Fairer Austausch

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades
eines Doktor-Ingenieurs (Dr.-Ing.)

von

Dipl.-Inform. Holger Vogt

aus Hofheim am Taunus

Referenten:	Prof. Dr. J. Buchmann
	Prof. Dr. R. Grimm
Datum der Einreichung:	28.04.2003
Datum der mündlichen Prüfung:	13.06.2003

Darmstadt 2003
D17

Danksagung

Zunächst einmal möchte ich allen danken, die mich während der Arbeit an dieser Dissertation unterstützt und begleitet haben. Hervorheben möchte ich die Unterstützung durch meine Eltern, die durch ihren bedingungslosen Rückhalt diese Dissertation ermöglicht haben.

Besonderer Dank gilt natürlich Prof. Dr. Johannes Buchmann, der diese Arbeit mit all seiner Erfahrung betreut und gefördert hat. Prof. Dr. Rüdiger Grimm danke ich dafür, daß er die Zweitbegutachtung meiner Dissertation übernommen hat.

Während meiner Arbeit hatte ich die Gelegenheit, mit zahlreichen Kollegen zusammenzuarbeiten. Dies sind insbesondere die Kollegen vom Graduiertenkolleg „Infrastruktur für den elektronischen Markt“, die Mitarbeiter des Lehrstuhls von Prof. Dr. Johannes Buchmann sowie die Mitarbeiter des (ehemaligen) Fachgebiets Betriebssysteme.

Besonders gefreut habe ich mich, wenn aus dieser Zusammenarbeit mit Kollegen auch Publikationen entstanden sind. Dankbar bin ich daher Felix Gärtner, Sigrid Gürgens, Dennis Kügler, Henning Pagnia, Carsten Rudolph und Uwe Wilhelm. Die Diskussionen und die gemeinsame Arbeit mit Felix Gärtner und Henning Pagnia, sowie meinem Reisegefährten Dennis Kügler habe ich dabei als besonders inspirierend empfunden.

Durch ein Stipendium der Deutschen Forschungsgemeinschaft (DFG) im Rahmen des Graduiertenkollegs „Infrastruktur für den elektronischen Markt“ war ich in der Lage, meiner Forschungsinteressen frei zu verfolgen.

Kurzfassung

Wenn zwei Parteien digitale Daten über eine Netzwerkverbindung austauschen, kann es zu Benachteiligungen kommen, falls eine Partei ihre Daten liefert und im Gegenzug nichts dafür erhält. Beim fairen Austausch wird dagegen sichergestellt, daß entweder beide Parteien etwas bekommen oder niemand etwas erhält. Auf diese Weise können Betrugsversuche beispielsweise beim Austausch digital signierter Verträge oder beim Einkauf digitaler Waren zuverlässig verhindert werden.

Zur Realisierung von fairem Austausch müssen jedoch spezielle Fairneßprotokolle eingesetzt werden, die auf die Dienste einer vertrauenswürdigen Partei zurückgreifen. Damit nicht jede Anwendung, die fairen Austausch verwenden will, eine eigene Infrastruktur für fairen Austausch betreiben muß, schlage ich einen generischen Fairneßdienst vor, der diese Aufgabe übernimmt. Wie ein solcher Dienst realisiert werden kann, ist das Thema dieser Arbeit. Dabei beschäftige ich mich zuerst mit der Spezifikation von fairem Austausch, dann mit der abstrakten Modellierung eines Fairneßdienstes und schließlich mit der Implementierung eines Prototyps.

Zuerst definiere ich eine Fairneßhierarchie, mit der sich die verschiedenen Fairneßeigenschaften eines Protokolls präziser als bisher beschreiben lassen. Dadurch kann besser zwischen verschiedenen Protokollen differenziert werden, was die Auswahl eines geeigneten Fairneßprotokolls erleichtert.

Danach stelle ich mein Konzept für einen Fairneßdienst vor, der den Austausch beliebiger digitaler Objekte unterstützt und verschiedene Fairneßprotokolle bereitstellt. Dieser Dienst bietet eine einheitliche Schnittstelle für alle praxisrelevanten Protokolle, so daß mit dieser Modellierung von den Details der Protokollimplementierung abstrahiert werden kann.

Anschließend diskutiere ich verschiedene Protokolle, die von meiner Modellierung eines Fairneßdienstes unterstützt werden. Neben einigen bereits bekannten Protokollen handelt es sich dabei auch um eine Reihe neu entwickelter Protokolle.

Dann untersuche ich verschiedene Probleme, die bei der prototypischen Implementierung meines Fairneßdienstes zu lösen waren. Dabei handelt es sich um Sicherheitsfragen, ebenso wie die Anforderung, möglichst jede beliebige Anwendung beim fairen Austausch unterstützen zu können. Schließlich zeige ich anhand von verschiedenen Beispielen, wie der generische Fairneßdienst in diversen Szenarien eingesetzt werden kann.

Einen alternativen Ansatz zu den in Software implementierten Fairneßprotokollen stellen in sicherer Hardware implementierte Protokolle dar. In dieser Arbeit untersuche ich erstmals das Potential von Hardwareunterstützung für fairen Austausch und weise nach, daß sichere Hardware das Lösen bisher ungelöster Probleme ermöglicht.

Abstract

If two parties exchange digital data by a network, one party risks to be disadvantaged. This may happen, if one party provides its data and receives nothing in exchange for it. This can be prevented by fair exchange which ensures that either both parties gain what they expect or nobody receives anything useful. Examples for the application of fair exchange are the signing of digital contracts and the purchase of digital goods.

Fair exchange can only be achieved with fair exchange protocols that rely on a trusted third party. As not every application that requires fair exchange can operate its own infrastructure including a trusted third party, I propose a generic fairness service that provides such an infrastructure. This thesis is about how such a fairness service can be realized. In more detail, this thesis covers a definition of fairness, an abstract model for a fairness service and its implementation issues.

First, I define a hierarchy of fairness levels that enables a more precise classification of fairness properties. This allows to differentiate more exactly between fairness protocols, which supports the selection of an appropriate fairness protocol.

Then, I present my concept for a fairness service that offers different fair exchange protocols to support fair exchange of digital items. This service abstracts from implementation details and provides a unified interface which covers all practical protocols.

Subsequently, I survey different protocols that are supported by my model for a fairness service. This model covers well known protocols as well as newly developed ones. These novel protocols for fair exchange especially exploit the item property of being revocable by the trusted third party.

Furthermore, I analyze different problems that have to be solved for the implementation of a prototype. Among these problems are appropriate security mechanisms and the demand to support arbitrary applications that require fair exchange. Several examples illustrate how my generic fairness service can be applied to concrete scenarios.

In addition to software implementations of fair exchange secure hardware can be used to execute fair exchange protocols. For the first time, this thesis investigates the potential of secure hardware to support fair exchange. My results provide the solution to an open problem: Secure hardware enables fair exchange of arbitrary digital items, even if the trusted third party is involved in conflict resolution only.

Erklärung

Hiermit erkläre ich, daß ich diese Dissertation – abgesehen von den in ihr ausdrücklich genannten Hilfen – selbständig verfaßt habe.

Wissenschaftlicher Werdegang des Verfassers in Kurzfassung

10/1993 - 12/1998	Studium Diplom-Informatik mit Nebenfach Mathematik an der TU Darmstadt
30.11.1998	Diplomabschluß (Dipl.-Inform.)
01/1999 - 04/2003	Promotionsstudent im Graduiertenkolleg „Infrastruktur für den elektronischen Markt“ an der TU Darmstadt

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

In dieser Arbeit entwickle ich eine flexible Infrastruktur für fairen Austausch. Ich entwerfe und implementiere einen Fairneßdienst, der von beliebigen Anwendungen zur Durchführung von fairem Austausch zwischen zwei Parteien genutzt werden kann. Der Einsatz eines solchen Fairneßdienstes stellt sicher, daß beim Austausch wertvoller digitaler Objekte entweder alle am Austausch beteiligten Parteien die gewünschten Objekte erhalten oder niemand etwas bekommt. Von einem solchen *fairen Austausch* profitieren insbesondere Anwendungen wie Vertragsunterzeichnung im Internet (engl. contract signing), Empfangsbestätigung für eine E-Mail (engl. certified e-mail) oder der Einkauf elektronischer Waren (engl. fair purchase).

Der Einsatz von fairem Austausch gemäß der in dieser Arbeit verwendeten Definition ist jedoch grundsätzlich mit zusätzlichem Aufwand verbunden: Even und Yacobi [EY80] (sowie später mit einer anderen Beweisidee Pagnia und Gärtner [PG99]) haben gezeigt, daß zwei Parteien allein keinen fairen Austausch durchführen können. Stattdessen wird die Hilfe eines vertrauenswürdigen *Vermittlers* benötigt, um Fairneß zu gewährleisten. Der in dieser Arbeit vorgeschlagene Fairneßdienst unterstützt beliebige Anwendungen beim fairen Austausch, so daß sich die Kosten für den Betrieb des Dienstes zwischen den vielen verschiedenen Nutzern aufteilen lassen. Dadurch kann ein gemeinsam genutzter Fairneßdienst deutlich kostengünstiger betrieben werden, als wenn jede Anwendung ihre eigene Lösung implementiert.

Die Probleme, die bei der Realisierung einer Infrastruktur für fairen Austausch zu lösen sind, lassen sich in drei Bereiche gliedern:

- Spezifikation von Fairneß
- Modellierung eines Fairneßdienstes
- Implementierung eines Fairneßdienstes

Eine Spezifikation von Fairneß wird benötigt, damit präzise Aussagen über die Eigenschaften von Fairneßprotokollen gemacht werden können. Nur mit einer fein abgestuften Fairneßdefinition kann eine Anwendung den gewünschten Fairneßgrad festlegen bzw. eine Protokollbeschreibung den zugesicherten Fairneßgrad benennen. Daher schlage ich eine Fairneßhierarchie vor, mit der sich die Fairneßeigenschaften präziser spezifizieren lassen als mit anderen bisher bekannten Definitionen. Ein Beispiel für eine andere, besonders bewährte Definition von Fairneß stammt von Asokan [Aso98, ASW97a] und wird auch in [GJM99, SM99] verwendet. Es gibt jedoch Protokolle, bei denen diese Definition nicht

hinreichend präzise ist: Die in dieser Arbeit mit P4 und P5 bezeichneten Protokolle besitzen gemäß der Definition von Asokan die gleichen Fairneßeigenschaften, obwohl sie sich in der Terminierungseigenschaft wesentlich unterscheiden. Dieses Problem löst meine Fairneßhierarchie, da sich mit ihr der Unterschied zwischen den beiden Protokollen präzise spezifizieren läßt. Somit erlaubt die Fairneßhierarchie eine bessere Differenzierung zwischen verschiedenen Austauschprotokollen und erleichtert so die Auswahl aus den vom Fairneßdienst bereitgestellten Protokollen.

Bei der Modellierung des Fairneßdienstes berücksichtige ich alle Protokolle, die ohne weitere Systemannahmen wie Gerichtsverfahren oder spezielle Kommunikationskanäle Fairneß garantieren können. Daher umfaßt mein FlexiFair genannter Fairneßdienst sowohl *aktive Protokolle*, bei denen sich der Vermittler immer aktiv am Austausch beteiligt, als auch *optimistische Protokolle*, bei denen der Vermittler nur zur Konfliktlösung im Fehlerfall benötigt wird. Durch die in dieser Arbeit vorgeschlagene modulare Modellierung von Fairneßprotokollen kann über eine einheitliche Schnittstelle auf aktive und optimistische Protokolle zugegriffen werden. Eine Anwendung kann sich daher bei der Auswahl eines Austauschprotokolls allein auf dessen Eigenschaften konzentrieren und muß die Art der Implementierung (aktiv oder optimistisch) nicht berücksichtigen.

Andere Artikel, die sich mit der Modellierung von generischen Fairneßprotokollen beschäftigen (z.B. [ASW98a, Aso98, LPSW00, Sch00, MS01]), konzentrieren sich dagegen auf optimistischen Austausch und ignorieren die aktiven Protokolle. Dies schränkt allerdings die Einsatzmöglichkeiten dieser generischen Protokolle stark ein, da die ausgetauschten Objekte bei optimistischen Protokollen immer *generierbar* oder *widerrufbar* sein müssen, d.h. der Vermittler muß die Objekte selbst erzeugen oder sie unbrauchbar machen können. Falls diese Eigenschaften den ausgetauschten Objekten fehlen, können sie nur mit aktiven Protokollen ausgetauscht werden. Daher ist es entscheidend für die vielseitige Einsetzbarkeit eines Fairneßdienstes, daß aktive Protokolle unterstützt werden. Nur auf diese Weise kann das von FlexiFair angestrebte Ziel erreicht werden, daß der Fairneßdienst eine Vielzahl von Anwendungen beim Austausch beliebiger Objekte unterstützt und somit maximale Flexibilität bietet.

Die bisher umfangreichste Modellierung eines Fairneßdienstes stammt von Schunter [Sch00] und ist auch in das SEMPER Projekt eingeflossen [SEM99, LPSW00]. Diese Modellierung umfaßt den optimistischen Austausch mit generierbaren als auch widerrufbaren Objekten. Allerdings wird dabei ein *synchrones Systemmodell* (siehe z.B. [Lyn96]) verwendet, d.h. Nachrichten müssen immer innerhalb einer festen Zeitspanne den Empfänger erreichen. In [ASW00, Abschnitt 2.2] wird jedoch dargelegt, daß die auf diesem Systemmodell basierenden Protokolle für den Austausch über das Internet nicht geeignet sind, da es bei der Festlegung der Zeitspanne für Nachrichtenlaufzeiten widersprüchliche Anforderungen gibt. Deshalb sind die von Schunter modellierten Protokolle kaum für den praktischen Einsatz geeignet. Stattdessen sollte immer ein *asynchrones Systemmodell* (siehe z.B. [Lyn96]) verwendet werden, das auf Zeitinformationen verzichtet und beliebige, endlich lange Nachrichtenlaufzeiten toleriert. Dieses auch bei meiner Modellierung verwendete Systemmodell vermeidet unnötige, kaum zu erfüllende Annahmen und entspricht somit am ehesten den in der Praxis gegebenen Verhältnissen.

Alle bisher bekannten optimistischen Fairneßprotokolle im asynchronen Systemmodell

(wie z.B. [GJM99, ZDB99, ASW00, KM00, MK01, MS01]) benötigen zwei generierbare Objekte und arbeiten immer nach dem gleichen, von Asokan vorgeschlagenen Konstruktionsprinzip [ASW98a, Aso98]. Daher ist es ein offenes Problem, ob mit widerrufbaren Objekten ein optimistisches Protokoll konstruiert werden kann, das die gleichen Fairneßeigenschaften besitzt, wie Asokans Protokoll mit zwei generierbaren Objekten. Alle bisherigen Arbeiten zu diesem Thema können entweder nur abgeschwächte Fairneßeigenschaften garantieren (z.B. [ASW98a, VP99, VPG99]) oder arbeiten im synchronen Systemmodell (z.B. [ASW97a, Sch00, BNS01]).

Als Lösung dieses Problems habe ich zwei Protokolle entworfen, die mit jeweils einem generierbaren und einem widerrufbaren Objekt arbeiten. Beide Protokolle nutzen die von mir vorgeschlagene Definition der starken und schwachen Generierbarkeit bzw. Widerrufbarkeit, die eine besonders präzise Beschreibung der Objekteigenschaften ermöglicht. Ich weise nach, daß diese beiden P3 und P4 genannten Protokolle mit stark widerrufbaren und schwach generierbaren Objekten bzw. schwach widerrufbaren und stark generierbaren Objekten den geforderten Grad der Fairneß erreichen. Für die Praxis ist besonders das Protokoll P3 mit dem stark widerrufbaren Objekt interessant, da es erstmals optimistisch fairen Austausch ohne starke Generierbarkeit erlaubt.

Ein weiterer Bereich, in dem diese Arbeit neue Ergebnisse erzielt, ist die Implementierung eines Fairneßdienstes. Als erster Punkt sind die verschiedenen Sicherheiteigenschaften auf Implementierungsebene zu nennen. Es ist wichtig, daß sich die gesendeten Nachrichten einer bestimmten Austauschtransaktion zuordnen lassen und daß das Wiedereinspielen oder Wiederverwenden von alten Nachrichten nicht möglich ist. Außerdem müssen sich die beteiligten Parteien authentisieren, damit unbefugte Parteien keinen Einfluß auf den Austauschvorgang nehmen können. Diese Sicherheitsanforderungen werden in den meisten Publikationen entweder ignoriert (z.B. [BP90, BF98, VPG99]) oder nur unzureichend behandelt (in [ZDB00, KM00, BK00, MK01] wurden z.B. verschiedene Angriffe übersehen). Daher zeige ich, wie sich durch den Einsatz einer richtig gewählten Transaktionsnummer die zuvor genannten Sicherheitsanforderungen erfüllen lassen. Diese Methode ist sehr generisch und läßt sich daher auf alle Austauschprotokolle und beliebige Objekte anwenden.

Eine andere Problemstellung bei der Implementierung ist die Erweiterbarkeit eines Fairneßdienstes. Wenn eine neue Anwendung Objekte austauschen möchten, die dem Vermittler unbekannt sind, muß es eine einfache Erweiterungsmöglichkeit für die Vermittlerfunktionalität geben. Da ein Vermittler aber besonders vertrauenswürdig sein soll, muß dessen Implementierung sehr sorgfältig durchgeführt werden, und es sollten möglichst wenige Änderungen oder Erweiterungen an dessen Software vorgenommen werden. Daher ist eine wie in [SEM99, Sch00] vorgesehene Erweiterbarkeit, bei der jede Anwendung neue Klassen definieren kann, die der Vermittler dann bei sich installiert, aus Sicherheits- sowie aus Kostengründen kaum durchführbar. Als Lösung für dieses Problem habe ich die dynamische Erweiterung von aktiven Protokollen entwickelt. Der Vermittler bekommt dabei zur Laufzeit die fehlenden Klassen geschickt und führt diese mit beschränkten Rechten aus, was in der Programmiersprache Java besonders einfach und sicher umsetzbar ist.

Schließlich zeige ich anhand von verschiedenen Beispielen, daß sehr unterschiedliche Anwendungen die von FlexiFair bereitgestellte Austauschfunktionalität nutzen können.

Die besondere Flexibilität meiner Modellierung und Implementierung zeigt sich dadurch, daß auch ein von mir neu vorgeschlagenes Austauschszenario problemlos durch die angebotenen Dienste unterstützt wird.

Im Gegensatz zu der bisher diskutierten Implementierung, die allein auf Software basiert, ist in Zukunft auch eine teilweise Implementierung von Austauschprotokollen in Hardware denkbar. Die immer schneller und trotzdem preiswerter werdenden Chipkarten bieten sich zur Realisierung einer solchen sicheren Hardwarelösung an. Daher untersuche ich in dieser Arbeit auch das Potential von Hardwareunterstützung für fairen Austausch. Es handelt sich dabei um die erste Arbeit, die sich mit diesem Szenario beschäftigt.

Ein Ergebnis in diesem Szenario bezieht sich auf *verderbliche Waren*. Dabei handelt es sich um Waren wie z.B. Echtzeitbörseninformationen, deren Wert mit der Zeit abnimmt. Der faire Austausch von verderblichen Waren wurde bereits von Asokan [Aso98, Seite 33] als ein offenes Problem genannt. Mit den in dieser Arbeit entwickelten Protokollen P7 und P8 gebe ich eine Lösung für dieses Problem an, die die besonderen Fähigkeiten von sicherer Hardware ausnutzt.

Ein weiteres Ergebnis betrifft die offene Frage [Aso98, Seite 137], ob beliebige digitale Objekte (d.h. Objekte, die nicht generierbar oder widerrufbar sind) optimistisch fair ausgetauscht werden können. Auch dieses Problem habe ich durch den Einsatz von sicherer Hardware gelöst. Dazu habe ich das Protokoll P9 entworfen, das die Vorteile von aktiven und optimistischen Protokollen in sich vereint. Es kann beliebige digitale Objekte austauschen und trotzdem handelt es sich um ein besonders effizientes optimistisches Protokoll. Aus diesen Gründen sind Protokolle mit Hardwareunterstützung den Softwareprotokollen deutlich überlegen und bieten sich in Zukunft als eine ideale Erweiterung eines Fairneßdienstes an.

Diese Arbeit ist wie folgt gegliedert:

In Kapitel 2 wird zuerst die Notwendigkeit eines Fairneßdienstes motiviert. Dann stelle ich nach den Systemannahmen die Fairneßhierarchie vor, die eine besonders fein abgestufte Beschreibung der Fairneßeigenschaften von Protokollen erlaubt. Bei der anschließenden Modellierung des Fairneßdienstes achte ich speziell darauf, daß aktive und optimistische Protokolle unterstützt werden und daß beliebige Objekte ausgetauscht werden können. Die wesentlichen Ideen dieses Kapitels wurden bereits in [VPG99, PVG03] veröffentlicht.

In Kapitel 3 werden alle wesentlichen Fairneßprotokolle vorgestellt und ihre Eigenschaften diskutiert. Bei den mit widerrufbaren Objekten arbeitenden Protokollen handelt es sich um teilweise in [Vog03] veröffentlichte Innovationen, die die Einsatzmöglichkeiten von optimistischen Protokollen deutlich erweitern. Um eine möglichst umfassende Zusammenstellung von Austauschprotokollen zu erreichen, werden in diesem Kapitel auch generische Protokollerweiterungen für Nichtabstreitbarkeit diskutiert.

In Kapitel 4 werden die Implementierungsdetails behandelt. Diese umfassen Lösungskonzepte für verschiedene Sicherheitsfragen [GRV03] ebenso wie das bereits erwähnte Konzept zur dynamischen Erweiterung der Vermittlerfunktionalität, das ich in [PVGW00] erstmals eingeführt habe. Des weiteren werden die Details der prototypischen Implementierung des Fairneßdienstes FlexiFair vorgestellt. Die vollständigen Schnittstellenbeschreibungen der wesentlichen Klassen sind jedoch wegen der besseren Übersicht-

lichkeit in den Anhang ausgelagert worden. Die verschiedenen Beispiele in diesem Kapitel zeigen schließlich, wie einfach und vielseitig sich dieser Fairneßdienst einsetzen läßt.

Kapitel 5 beschäftigt sich ausschließlich mit dem Einsatz von sicherer Hardware zur Unterstützung von fairem Austausch. Dieses neue Szenario habe ich auch in den Veröffentlichungen [VPG01, VGP03, PVG02] ausführlich untersucht. Es werden optimistische Protokolle für den Austausch von verderblichen Waren sowie für den fairen Austausch von beliebigen Objekten vorgestellt und deren Eigenschaften nachgewiesen. Damit sich solche Protokolle auch auf weniger leistungsfähiger Hardware ausführen lassen, schlage ich verschiedene Optimierungen vor, die den Speicher- und Rechenbedarf der Hardware reduzieren.

Kapitel 6 faßt schließlich die wesentlichen Ergebnisse dieser Arbeit zusammen.

2 Ein generischer Fairneßdienst

In diesem Kapitel stelle ich die Modellierung meines generischen Fairneßdienstes FlexiFair vor. Dieser unterstützt die Implementierung von fairem Austausch in vielen Anwendungen, wie z.B. faire Vertragsunterzeichnung, fairer Einkauf mit elektronischem Geld, etc. FlexiFair ist dabei so generisch entworfen, daß es diverse Austauschprotokolle, sowie viele auszutauschende Objekte unterstützt. Bei Bedarf kann FlexiFair um weitere Protokolle und neue digitale Objekte erweitert werden, wodurch es sich leicht an neue Anforderungen anpassen läßt.

In Abschnitt 2.1 wird zuerst motiviert, welche Vorteile ein generischer Fairneßdienst wie FlexiFair mit sich bringt. Um die Fairneßeigenschaften eines Protokolls präzise beschreiben zu können, schlage ich in Abschnitt 2.2 eine Fairneßhierarchie vor. Diese erlaubt eine bessere Bewertung von Lösungen für fairen Austausch, als dies die bisher vorgeschlagenen Begriffe ermöglichen.

In Abschnitt 2.3 stelle ich meine Konzepte für den Fairneßdienst FlexiFair vor, die das Ziel verfolgen, diverse Austauschprotokolle und eine Vielzahl von Objekten zu unterstützen. Das besondere an der vorgeschlagenen modularen Modellierung der Austauschprotokolle besteht darin, daß FlexiFair damit sowohl aktive als auch optimistische Protokolle unterstützen kann. Eine Anwendung kann über eine einheitliche Schnittstelle auf Austauschprotokolle zugreifen, ohne deren Implementierung berücksichtigen zu müssen. Trotzdem besitzt die Anwendung weiterhin die Möglichkeit, Einfluß auf den Protokollablauf zu nehmen, z.B. wenn nach einem Kommunikationsfehler über die Art der Konfliktlösung entschieden werden soll.

Schließlich führe ich die neuen Begriffe der starken und schwachen Generierbarkeit bzw. Widerrufbarkeit ein, die die Grundlage für die in Kapitel 3 neu entwickelten Protokolle bilden. Diese Protokolle demonstrieren dann auch die Leistungsfähigkeit der für FlexiFair entwickelten Konzepte. Die Implementierung dieser Konzepte wird schließlich in Kapitel 4 präsentiert.

2.1 Motivation

Im folgenden werden die Grundgedanken aus Kapitel 1 aufgegriffen und die Vorteile und Problemstellungen eines generischen Fairneßdienstes diskutiert. Zum Abkürzen der Lektüre könnte man diesen Abschnitt auch überspringen und bei der Fairneßdefinition in Abschnitt 2.2 weiterlesen.

2.1.1 Warum wird ein generischer Fairneßdienst benötigt?

Fairer Austausch ist ein wichtiger Basisdienst, der eine sichere und benachteiligungsfreie Ausführung von E-Commerce Anwendungen ermöglicht. Zu solchen Anwendungen, die fairen Austausch benötigen, zählen zum Beispiel Programme zur Vertragsunterzeichnung im Internet oder elektronische Zahlungssysteme. Heute setzen diese Anwendung größtenteils keinen fairen Austausch ein, da dies mit zusätzlichem Aufwand verbunden ist: Fairer Austausch ist nur mit einem vertrauenswürdigen Vermittler möglich [EY80, PG99]. Damit dieser Vermittler bei einem Austausch Fairneß garantieren kann, muß er praktisch immer verfügbar sein. Dadurch entstehen relativ hohe Kosten beim Vermittler, was dessen Betrieb oft unwirtschaftlich macht und so den Einsatz von fairem Austausch verhindert. Insbesondere bei einer Anwendung mit wenigen Nutzern bzw. wenigen Austauschvorgängen stellt fairer Austausch und der damit verbundene Einsatz eines Vermittlers einen nicht unerheblichen Kostenfaktor dar, der dem breiten Einsatz von fairem Austausch entgegensteht.

Diese Kosten lassen sich jedoch deutlich senken, wenn mehrere unterschiedliche Anwendungen gemeinsam einen Fairneßdienst verwenden und deshalb auf den gleichen Vermittler zugreifen können. Dann verteilen sich die Fixkosten für den Betrieb des Vermittlers auf viele Nutzer bzw. Austauschvorgänge, wodurch der Einsatz von fairem Austausch erst rentabel und somit praktikabel wird. Ein generischer Fairneßdienst, der eine einheitliche Infrastruktur für fairen Austausch zur Verfügung stellt, löst daher das Problem, wie fairer Austausch möglichst kostengünstig in eine Vielzahl von E-Commerce Anwendungen integriert werden kann.

Ein generischer Fairneßdienst hat auch bezüglich der Sicherheit einige Vorteile: Da in der Vergangenheit immer wieder neue Angriffe auf Fairneßprotokolle entdeckt wurden (z.B. [ZDB00, BK00, KR01]), muß auch in Zukunft damit gerechnet werden, daß viele Protokollimplementierungen nicht fehlerfrei sind.

Wenn jede Anwendung einen selbstentwickelten Fairneßdienst betreibt, der typischerweise stark spezialisierte Fairneßprotokolle verwendet, dann muß jeder Betreiber die Sicherheit seines Fairneßdienstes auch selbst überwachen. Dies erfordert jedoch eine gute Sachkenntnis, die meist nicht vorhanden oder nur mit hohen Kosten zu gewährleisten ist. Wenn dagegen bei einem generischen Fairneßdienst die gleichen Protokolle von verschiedenen Anwendungen genutzt werden, dann rechtfertigt dies einen höheren Aufwand für der Wartung und Pflege der Software durch Sicherheitsexperten. In diesem Sinne kann der Sicherheitsvorteil auch wieder unter Einsparung von Kosten verbucht werden.

Ein weiterer großer Vorteil eines generischen Fairneßdienstes besteht darin, daß eine Anwendung das Austauschprotokoll auswechseln kann, sobald ein Fehler im bisher verwendeten Protokoll entdeckt wird. Da alle Protokolle über die gleiche Schnittstelle angesprochen werden, ist dazu keine Änderung der Anwendung notwendig: Es genügt den neuen Protokollnamen anzugeben, um von einem fehlerhaften auf ein fehlerfreies Protokoll umzuschalten.

2.1.2 Welche Probleme müssen für die Realisierung eines Fairneßdienstes gelöst werden?

Heute bereits bestehende Dienste, die fairen Austausch unterstützen, sind zu unflexibel, um von mehreren Anwendungen genutzt zu werden. Üblicherweise wird nur ein einziges Austauschprotokoll implementiert, welches wiederum nur genau eine bestimmte Anwendung unterstützt (z.B. nur ein bestimmtes elektronisches Zahlungssystem oder nur E-Mail Empfangsbestätigungen [Cer02, Rea02]). Das Netbill Zahlungssystem [CTS95] enthält zum Beispiel einen Mechanismus für den fairen Austausch von Geld gegen den Dechiffrierschlüssel der verschlüsselten Ware. Eine weitere Überprüfung der Ware, z.B. ob sie eine gültige Signatur enthält, ist dabei nicht mehr automatisch möglich, so daß andere Einsatzgebiete wie der Austausch von Signaturen ausgeschlossen sind. Außerdem wird nur ein einziges Austauschprotokoll verwendet, was ebenfalls einer anderweitigen Verwendung entgegensteht.

Im Gegensatz dazu stellt mein generischer Fairneßdienst diverse Austauschprotokolle bereit und unterstützt den Austausch vieler unterschiedlicher Objekte. Beim Entwurf eines solchen Dienstes gibt es eine Reihe von Fragen zu beantworten:

Fairneßgrad von Protokollen: Welche Eigenschaften müssen Protokolle für den fairen Austausch erfüllen? Welche optionalen Eigenschaften können Fairneßprotokolle noch haben? Wie kann der Fairneßgrad von verschiedenen Protokollen verglichen werden?

Eigenschaften der ausgetauschten Objekten: Was sind die wesentlichen Eigenschaften von austauschbaren Objekten? Welche Eigenschaften werden für eine erfolgreiche Konfliktlösung in Austauschprotokollen benötigt? Welche Protokolle lassen sich mit diesen Objekteigenschaften überhaupt realisieren?

Protokolle mit Vermittler im Fehlerfall vs. aktive Protokolle: Welche Gemeinsamkeiten haben Protokolle, bei denen der Vermittler nur im Fehlerfall beteiligt ist, und Protokolle mit aktiver Beteiligung des Vermittlers? Wie läßt sich eine einheitliche Schnittstelle für beide Arten von Protokollen schaffen?

Antworten auf diese Fragen sind wichtig, damit Anwendungen immer auf die gleiche Art und Weise auf die verschiedenen Austauschprotokolle zugreifen können. Auch mit anderen Objekten sollen der Austausch dann genauso gut einsetzbar bleiben.

Die in Kapitel 4 beschriebenen Details der prototypischen Implementierung von FlexiFair werfen weitere Problemen auf:

Sicherheit: Wie authentisieren sich die am Austausch beteiligten Parteien? Wie werden Nachrichten der Parteien einer bestimmten Austauschtransaktion zugeordnet?

Erweiterbarkeit: Wie kann ein Fairneßdienst um weitere Protokolle ergänzt werden? Kann eine Partei eigene Objekte und deren Beschreibungen definieren?

Effizienz: Wie effizient sind verschiedene Protokolle? Welchen Einfluß haben die verwendeten Objekte und deren Beschreibungen?

Erst durch die Antworten auf diese Fragen wird schließlich in Kapitel 4 die Praktikabilität des in diesem Kapitel vorgestellten Konzepts für einen Fairneßdienst unter Beweis gestellt.

2.2 Fairneß

In diesem Abschnitt stelle ich meine Definition einer Fairneßhierarchie vor. Diese bildet die Grundlage für die Bewertung der im Rahmen des Fairneßdienstes FlexiFair bereitgestellten Protokolle. Im folgenden wird zuerst die Notwendigkeit für eine allgemein verwendbare Fairneßhierarchie begründet, und die Systemannahmen werden festgelegt. Dann werden in Abschnitt 2.2.3 die verschiedenen Protokolleigenschaften der Fairneßhierarchie spezifiziert und in Abschnitt 2.2.4 mit anderen Fairneßdefinitionen verglichen.

2.2.1 Einleitung

Bei vielen bisher vorgeschlagenen Fairneßdefinitionen besteht das Problem, daß sie nur für bestimmte Klassen von Austauschprotokollen entwickelt wurden. Zum Beispiel wurde die Definition von „schwacher Fairneß“ von Asokan [Aso98, ASW97a] speziell für Protokolle entwickelt, bei denen ein vertrauenswürdiger Vermittler lediglich im Fehlerfall eingeschaltet wird. Tygar's Fairneßbegriffe „money atomicity“, „goods atomicity“ und „certified delivery“ [Tyg96, CHTY96, Tyg98] beziehen sich dagegen nur auf den Austausch von Ware gegen Geld und eignen sich aufgrund ihrer Gemeinsamkeiten mit Datenbanktransaktionen eher für Austauschprotokolle mit aktiv beteiligtem Vermittler.

Im Gegensatz dazu stelle ich im folgenden eine Fairneßhierarchie vor, die fein abgestufte Grade der Fairneß definiert. Dadurch lassen sich die Fairneßeigenschaften der meisten Austauschprotokolle exakt spezifizieren. Eine solche Spezifikation ist eine wichtige Voraussetzung für den praktischen Einsatz einer Protokollimplementierung, denn die Auswahl eines Protokolls muß anhand von exakten Aussagen über den Grad der Fairneß getroffen werden. Somit bildet die hier vorgestellte Fairneßhierarchie die Grundlage für den Einsatz von Austauschprotokollen im Fairneßdienst FlexiFair.

Meine Fairneßhierarchie basiert auf der Fairneßdefinition von Asokan [Aso98] (siehe auch Abschnitt 2.2.4) und erweitert diese um deutlich feinere Abstufungen beim Grad der Fairneß und der Terminierung. Deshalb kann diese Definition einer Fairneßhierarchie für eine Vielzahl von Anwendungen und Protokollen eingesetzt werden.

2.2.2 Systemannahmen

Die Definition der Fairneßhierarchie setzt die folgenden Systemannahmen als gegeben voraus:

Es gibt zwei Parteien A und B, die ihre Objekte O_A und O_B austauschen wollen. Beide Parteien besitzen eine *Beschreibung* der Eigenschaften des Objekts, das sie während des Austausches erhalten möchten. Anhand dieser Beschreibung sind sowohl A und B als auch jede andere Partei in der Lage, die Übereinstimmung eines empfangenen Objekts mit der Beschreibung zu verifizieren.

Die Kommunikation zwischen A und B, sowie mit jeder anderen Partei erfolgt allein durch das Versenden von Nachrichten über ein Kommunikationsmedium. Eine Nachricht kann unterwegs verloren gehen (z.B. durch eine Störung des Kommunikationsmediums) oder aber ihren Empfänger erreichen. Der Sender einer Nachricht kann in der Regel nicht erkennen, ob seine Nachricht verloren gegangen ist, ob sie noch unterwegs ist oder ob sie den Empfänger erreicht hat. Ebenso kann ein Empfänger nicht unterscheiden, ob eine Nachricht nicht gesendet wurde oder ob sie nur noch nicht eingetroffen ist. Diese Art der Kommunikation wird üblicherweise als *asynchrone Kommunikation* bezeichnet (siehe z.B. [Sch93] oder [Lyn96]). Die in manchen Protokollen vorausgesetzte *synchrone Kommunikation* [Sch93, Lyn96], bei der Nachrichten immer innerhalb einer festen Zeitspanne ankommen, wird in dieser Arbeit nur am Rande betrachtet, da es sich nicht um ein gutes Modell für die heutige Internetkommunikation handelt (siehe auch [ASW00, Abschnitt 2.2] für eine ausführliche Diskussion der Nachteile von synchroner Kommunikation).

Damit fairer Austausch innerhalb dieses Systemmodels überhaupt lösbar ist, wird immer ein vertrauenswürdiger *Vermittler* V benötigt (Beweis siehe [EY80, PG99]). Dieser ist unparteiisch, arbeitet fehlerfrei das vorgegebene Protokoll ab und ist ständig erreichbar. Grundsätzlich wird der Vermittler erst nach einer Anfrage von A oder B aktiv, d.h. er reagiert lediglich auf Anfragen und wird nicht von sich aus aktiv. Für die Kommunikation mit dem Vermittler muß die zusätzliche Annahme gelten, daß Nachrichten an ihn bzw. von ihm nicht verloren gehen und daher immer ankommen. Durch diese Annahme (auch „resilient channel“ genannt [ASW98a]) wird sichergestellt, daß jede Partei irgendwann eine Antwort auf ihre Anfrage beim Vermittler erhalten wird. Im Gegensatz dazu kann die Kommunikation zwischen A und B sogar dauerhaft gestört sein bzw. absichtlich für immer unterbrochen werden.

Die Parteien A und B können sich jederzeit an den Vermittler wenden, wenn ein Fehler beim Austausch der Objekte aufgetreten ist. Der Vermittler überprüft dann, ob es sich wirklich um einen Fehler handelt, und versucht einen vorhandenen Fehler zu korrigieren. Diesen Teil eines Protokolls bezeichne ich im folgenden mit dem Begriff *Konfliktlösung*. Das Ergebnis einer solchen Konfliktlösung teilt der Vermittler schließlich der aufrufenden Partei in seiner Antwort mit. Der Vermittler muß seine Entscheidung speichern, damit er bei einer erneut gestarteten Konfliktlösung immer eine konsistente Antwort liefert.

Ein *Gericht* wird in dieser Arbeit als eine Erweiterung des Vermittlers angesehen. Ein Gericht ist nie am Austausch direkt beteiligt, sondern wird nur für den nicht automatisierbaren Teil der Konfliktlösung eingesetzt. Während der Vermittler alle Aktionen automatisiert durchführt, entscheidet bei einem Gerichtsverfahren ein Richter individuell über die vorgebrachten Argumente. Dabei kann ein Richter auch eine Befragung der beteiligten Parteien durchführen oder die Übereinstimmung eines Objekts mit der geforderten Beschreibung durch einen Gutachter prüfen lassen. Auch zur Durchsetzung einer Gerichtsentscheidung existieren mehr Möglichkeiten als bei einer Konfliktlösung mit Vermittler. Ein Gericht kann dazu verschiedene Zwangsmaßnahmen verhängen wie zum Beispiel Gefängnisstrafen, Geldstrafen, Beschlagnahmen von Beweisen, etc. Trotzdem gibt es auch für das Gericht Grenzen bei der Durchsetzung von Fairneß: Nur wenn Beweise vorliegen, die Auskunft über den bisherigen Ablauf des Austauschprotokolls geben, kann

ein Gericht immer eine korrekte Entscheidung treffen. Ohne diese Beweise besteht die Gefahr, daß Gerichtsentscheidungen inkonsistent zum bisherigen Protokollablauf sind, so daß ein Gerichtsverfahren in diesem Fall keine Fairneß garantieren kann.

Ein *Austauschprotokoll* beschreibt das Verhalten von A, B und V. Diese Parteien können die folgenden Aktionen ausführen:

- Nachrichten senden oder empfangen
- Berechnungen durchführen
- Terminieren (d.h. keine weiteren Aktionen mehr ausführen)
- ein Gerichtsverfahren starten

Da ein Gerichtsverfahren deutlich höhere Kosten verursacht als die anderen automatisierten Aktionen, beschränke ich mich in dieser Arbeit nach Möglichkeit auf Austauschprotokolle, in denen kein Gerichtsverfahren vorgesehen ist.

Zu jedem Zeitpunkt besitzen die Parteien einen *Zustand*. Für die Partei A genügt es, die folgenden beiden Zustände zu unterscheiden:

1. „A hat O_B“ oder
2. „A hat O_B nicht“.

Die Zustände der Partei B sind analog definiert.

Eine *Zustandsänderung* der Parteien A und B kann dann nur durch eine der folgenden Aktionen ausgelöst werden:

- Die Partei empfängt eine Nachricht mit dem gewünschten Objekt.
- Der Vermittler führt eine Aktion aus, die den Zustand von A oder B unmittelbar verändert. Eine solche Aktion ist allerdings abhängig von den ausgetauschten Objekten. Beispielsweise kann der Vermittler die Rückbuchung einer Zahlung von A an B veranlassen, so daß B das Objekt O_A nicht mehr hat. Umgekehrt könnte der Vermittler die Zahlung auch direkt auf das Konto von B einzahlen, so daß dieser O_A unmittelbar erhält.
- Ein Gericht kann genauso wie der Vermittler Aktionen durchführen, die den Zustand von A oder B ändern.

Die zustandsverändernden Aktionen des Vermittlers und des Gerichts schließen sich gegenseitig aus: Wenn der Vermittler den Zustand einer Partei verändert, dann darf das Gericht diese Entscheidung nicht mehr durch eine weitere Zustandsänderung revidieren. Ebenso darf der Vermittler bzw. das Gericht seine eigene Entscheidung für eine Zustandsänderung nicht mehr revidieren. In einem Austauschprotokoll werden also der Vermittler und das Gericht den Zustand einer Partei höchstens einmal verändern. Durch diese Einschränkungen wird vermieden, daß sich die Aussagen des Vermittlers bzw. des Gerichts über den Ausgang des Austausches widersprechen.

Die zustandsverändernden Aktionen des Vermittlers bzw. des Gerichts sind endgültig: Nach einer solchen Aktion kann der Empfang einer Nachricht keine Zustandsänderung mehr auslösen. Entweder die Partei besitzt das Objekt schon oder es wurde ihr vorher weggenommen, wodurch auch die in Nachrichten verschickten Kopien des Objekts ungültig geworden sind. Am Beispiel der oben genannten Zahlung bedeutet dies, daß das vom Vermittler auf das Konto von B eingezahlte Geld nur einmal von der Bank gutgeschrieben wird, beziehungsweise daß eine stornierte Zahlung nicht doch noch dem Konto gutgeschrieben werden kann.

Da das mehrmalige Empfangen eines Objekts nur beim ersten Mal eine Zustandsänderung der Partei hervorrufen kann, folgt aus diesen Überlegungen, daß sich der Zustand einer Partei während eines Austauschprotokolls höchstens zwei Mal ändert. Diese Eigenschaft gilt unabhängig davon, ob sich eine Partei an das Protokoll hält oder nicht.

2.2.3 Definition einer Fairneßhierarchie

Ein Protokoll für den *fairen Austausch* sollte immer die folgenden vier Eigenschaften besitzen: Wirksamkeit, Terminierung, Fairneß und Nichtabstreitbarkeit. Für eine konkrete Anwendung können eventuell noch weitere optionale Eigenschaften gefordert werden, um jede Art der Benachteiligung einer Partei auszuschließen. Der genaue Grad der Fairneß hängt dann davon ab, in welchem Umfang die einzelnen Eigenschaften erfüllt sind.

1. Wirksamkeit

Durch die Protokolleigenschaft der Wirksamkeit wird sichergestellt, daß ein Protokoll immer versucht, einen Austausch durchzuführen. Insbesondere die triviale Lösung, einfach nichts auszutauschen, ist dadurch ausgeschlossen.

Wirksamkeit: Wenn keine Kommunikationsfehler auftreten, A und B sich an das Protokoll halten und nicht den vorzeitigen Abbruch des Protokolls fordern, dann erhalten beide Parteien das von ihnen gewünschte Objekt.

2. Terminierung

Diese Protokolleigenschaft stellt die Beendigung des Austauschprotokolls durch die Partei $P \in \{A, B\}$ sicher. Die jeweils andere Partei wird dann Q genannt.

Terminierung für P: Wenn sich P an das Protokoll hält, dann wird P die Ausführung des Protokolls beenden.

Wenn ein Protokoll diese Terminierungseigenschaft für die Partei P garantiert, so gilt dies unabhängig davon, wie sich die Partei Q verhält. Auch wenn Q fehlerhafte Nachrichten schickt oder die Kommunikationsverbindung zu P dauerhaft unterbricht, wird das Protokoll immer die Terminierung der Partei P sicherstellen.

Eine weitere Präzisierung dieser Terminierungsbedingung ergibt sich, wenn auch die Frage betrachtet wird, ob sich der Zustand einer Partei (entweder „P hat O_Q“ oder „P

hat O_Q nicht“) nach der Terminierung noch ändern kann. Ein solcher Fall kann dann auftreten, wenn der Vermittler nach der Terminierung von P doch noch das gewünschte Objekt liefert oder es unbrauchbar macht.

Die folgenden drei Terminierungseigenschaften eines Protokolls sind dann für die Partei $P \in \{A, B\}$ möglich:

T2: Wenn sich P an das Protokoll hält, dann terminiert P und nach der Terminierung ändert sich der Zustand von P nicht mehr.

T1: Wenn sich P an das Protokoll hält, dann terminiert P und nach der Terminierung kann sich der Zustand von P durch eine Aktion des Vermittlers noch einmal ändern.

Dabei lassen sich insbesondere die folgenden beiden Fälle von T1-Terminierung unterscheiden:

T1+ Der Zustand von P kann sich nach der Terminierung von „P hat O_Q nicht“ auf „P hat O_Q“ ändern.

T1− Der Zustand von P kann sich nach der Terminierung von „P hat O_Q“ auf „P hat O_Q nicht“ ändern.

T0: Wenn sich P an das Protokoll hält, dann kann P nicht in jedem Fall terminieren.

Die Terminierungseigenschaft T0 ist hier nur der Vollständigkeit halber aufgeführt. Es handelt sich dabei um den Fall, daß für eine Partei keine Terminierung möglich ist. Bei einem Austauschprotokoll sollte daher immer mindestens T1 oder T2 für die Parteien A und B gelten, so daß es für diese zwei Parteien nur 4 sinnvolle Kombinationen der Terminierung gibt.

Die Terminierung der Form T1 ist auch nicht immer ausreichend: Eine Partei terminiert bei T1 zwar auf jeden Fall, aber der Zustand dieser Partei kann sich noch ändern. Im Fall T1+ wird ein Austausch möglicherweise später doch noch erfolgreich beendet, was in manchen Fällen — wie z.B. ein für den Händler unerwartet doch noch beendeter Einkauf — keine nachteilige Form der Terminierung ist. Bei T1− muß dagegen damit gerechnet werden, daß der Austausch später noch zurückgesetzt wird. Damit reduziert diese Form der Terminierung den Nutzen des bereits empfangenen Objekts, da es möglicherweise seinen Wert zu einem beliebigen späteren Zeitpunkt verliert. Daher ist die Terminierung T1− in der Praxis meist nicht viel besser als T0.

Trotzdem bietet die T1-Terminierung neben der Möglichkeit zu terminieren einen weiteren klaren Vorteil gegenüber der T0-Terminierung: Da der Vermittler den Zustand von P höchstens einmal ändern wird, kennt P das endgültige Resultat des Austausches, sobald der Vermittler diese Aktion durchführt.

3. Fairneß

Um die Fairneß eines Protokolls zu beschreiben, muß zuerst die *Fairneß eines Protokollzustands* für eine Partei $P \in \{A, B\}$ definiert werden. Die jeweils andere Partei wird wieder Q genannt.

Fairer Protokollzustand für P: Die Partei P hat das gewünschte Objekt O_Q oder Q hat das Objekt O_P nicht.

Wenn dagegen ein Zustand für P nicht fair ist (d.h. P hat O_Q nicht und Q hat O_P), dann nennt man dies auch einen *unfairen Protokollzustand*.

Die *Fairneßeigenschaft eines Protokolls* ist für eine Partei $P \in \{A, B\}$ wie folgt definiert:

Fairneß für Partei P: Wenn sich P an das Protokoll hält und terminiert, dann befindet sich P ab dem Terminierungszeitpunkt immer in einem fairen Protokollzustand. Wenn sich P an das Protokoll hält und nicht terminiert, dann gilt ab der letzten Zustandsänderung von P und Q, daß sich P in einem fairen Protokollzustand befindet.

Diese Fairneßdefinition stellt sicher, daß ein Protokoll, das Fairneß für Partei P garantiert, immer in einem für P fairen Zustand endet. Dies gilt unabhängig von der Terminierungseigenschaft: Falls P terminiert, so befindet er sich schon ab diesem Zeitpunkt in einem fairen Zustand. Falls P nicht terminiert, so wird ein fairer Zustand spätestens bei der für beide Parteien letzten Zustandsänderung erreicht. Die Überlegungen am Ende des Abschnitts 2.2.2 haben gezeigt, daß es höchstens zwei Zustandsänderungen pro Partei geben kann. Daher gibt es immer eine letzte Zustandsänderung der Parteien. Falls sich die Zustände beider Parteien während der Protokollausführung nicht ändern, so ist der Zeitpunkt der letzten Zustandsänderung identisch mit dem Start des Protokolls.

Genau wie bei der Terminierung gilt auch hier, daß selbst eine betrügerische Partei Q die Fairneß für eine ehrlich Partei P nicht verhindern kann.

Die vollständige Fairneßeigenschaft des Protokolls kann schließlich wie folgt definiert werden:

Fairneß eines Protokolls: Ein Protokoll ist *fair*, wenn es Fairneß für A und B garantiert.

Der Nachweis dieser Fairneßeigenschaft kann unter verschiedenen Voraussetzungen geführt werden, was zu unterschiedlich starken Fairneßgarantien führt: Falls bereits schwache Annahmen für die Realisierung von fairem Austausch ausreichen, so ist eine solche Fairneßgarantie stärker als wenn schwer zu erfüllende Annahmen vorausgesetzt werden.

Die wichtigste Unterscheidung besteht darin, ob Fairneß automatisch durch Austauschprotokolle garantiert werden kann oder ob zum Erreichen der Fairneß möglicherweise auch ein Gerichtsverfahren notwendig ist. Dabei bezeichne ich als eine Realisierung von Fairneß *innerhalb des Systems* genau den Fall, daß die Fairneß allein durch das Versenden von Nachrichten zwischen den Parteien A, B und V sowie durch Berechnungen dieser Parteien erreicht werden kann. Dagegen greift eine Realisierung von Fairneß *außerhalb des Systems* bei einem Gerichtsverfahren auf zusätzliche, nicht-elektronische Maßnahmen zurück.

Eine weitere Unterscheidungsmöglichkeit für die Stärke der Fairneßgarantie liefert die Frage, ob sich beide Parteien, nachdem ein Fehler aufgetreten ist, an der Konfliktlösung mit dem Vermittler beteiligen werden. Bei einer von P gestarteten Konfliktlösung kann

der Vermittler nicht unbegrenzt lange auf Nachrichten von Q warten. Daher muß bei einer solchen Konfliktlösung eine feste Zeitschranke für die Nachrichtenlaufzeiten bei der Kommunikation mit Q existieren, was durch synchrone Kommunikation (siehe S. 11) sichergestellt wird. Schunter [Sch00, S. 206] nennt diese Mitwirkung aller Parteien an der Konfliktlösung auch Drei-Parteien Konfliktlösung (engl. three-party dispute).

Die aus diesen verschiedenen Voraussetzungen resultierenden *Fairneßgrade* lassen sich dann wie folgt anordnen:

F4: Fairneß für die Partei P wird unter den folgenden Voraussetzungen durch das Austauschprotokoll (also innerhalb des Systems) garantiert:

- Der Vermittler ist ständig verfügbar und Nachrichten an ihn und von ihm gehen nicht verloren.

F3: Fairneß für die Partei P wird unter den folgenden Voraussetzungen durch das Austauschprotokoll (also innerhalb des Systems) garantiert:

- Es gelten die gleichen Voraussetzungen wie bei F4 und zusätzlich
- existiert ein synchroner Kommunikationskanal zwischen dem Vermittler und Q, so daß der Vermittler auch Q an der automatischen Konfliktlösung beteiligen kann.

F2: Fairneß für die Partei P wird unter den folgenden Voraussetzungen garantiert:

- Es gelten die gleichen Voraussetzungen wie bei F4 und zusätzlich
- kann eine Konfliktlösung mittels eines Gerichtsverfahrens durchgeführt werden.

F1: Fairneß für die Partei P wird unter den folgenden Voraussetzungen garantiert:

- Es gelten die gleichen Voraussetzungen wie bei F3 und zusätzlich
- kann eine Konfliktlösung mittels eines Gerichtsverfahrens durchgeführt werden.

F0: Fairneß für die Partei P kann nicht garantiert werden.

Die Stärke der Fairneßgarantien sind von F4 nach F0 abnehmend: Während bei F4 und F3 das Austauschprotokoll immer selbst automatisch Fairneß erreichen kann, muß bei F2 und F1 möglicherweise auf Gerichtsverfahren und somit auf Maßnahmen außerhalb des Systems zurückgegriffen werden, die nicht im Einflußbereich des Austauschprotokolls liegen. Da bei einem Gerichtsverfahren der Ablauf typischerweise nicht genau vorhersagbar ist, bieten F2 und F1 nach meiner Einschätzung eine schwächere Fairneßgarantie. Ein klarer Nachteil dieser nicht automatisierten Lösungen besteht darin, daß sie mit deutlich höheren Kosten und größerem Zeitaufwand verbunden sind. Trotzdem läßt sich ein

Gerichtsverfahren nicht immer vermeiden: Falls z.B. bei einem Objekt nicht automatisiert entschieden werden kann, ob es seiner Beschreibung entspricht, muß dies von einem Gutachter außerhalb des Systems entschieden werden.

Die Definition von F0 ist nur der Vollständigkeit halber aufgeführt, damit sich auch nicht faire Austauschprotokolle in diese Hierarchie einordnen lassen. F0 bzw. keine Fairneß erreichen zum Beispiel Protokolle, die ohne Vermittler arbeiten (siehe dazu auch Abschnitt 2.3.2).

Die hier definierte Fairneßhierarchie läßt sich problemlos durch das Hinzufügen oder Weglassen von Voraussetzungen erweitern. Man könnte zum Beispiel bei F4 die Forderung weglassen, daß Nachrichten zum und vom Vermittler nicht verloren gehen. Weil dann Fairneß unter geringeren Voraussetzungen garantiert wird, handelt es sich um einen stärkeren Grad der Fairneß, den man F5 nennen könnte. Es gibt allerdings kein Protokoll, das diese F5-Fairneß für beide Parteien garantiert. Bei F5 könnten alle Nachrichten zum und vom Vermittler verloren gehen, so daß ein Protokoll Fairneß ohne den Vermittler erreichen müßte, was — wie bereits erwähnt — grundsätzlich nicht möglich ist [EY80]. Daher beschränke ich mich auf F4–F0, womit alle in der Praxis relevanten Protokolle in die Fairneßhierarchie eingeordnet werden können.

Die für einzelne Parteien definierten Fairneßlevel F4–F0 verwende ich auch zur Charakterisierung der Fairneß von Protokollen. Der Grad der Fairneß eines Protokolls entspricht dann dem Minimum der den Parteien garantierten Fairneßgrade. Wenn ein Protokoll zum Beispiel F4-Fairneß für A und F2-Fairneß für B garantiert, dann ist das Protokoll F2-fair. Welchen Grad der Fairneß ein Protokoll einer Partei bietet, kann auch besonders einfach anhand der Fragen in Abbildung 2.1 entschieden werden.

4. Nichtabstreitbarkeit

Ein Fairneßprotokoll kann den beteiligten Parteien auch *Nichtabstreitbarkeit* (engl. nonrepudiation [ISO97]) bieten, d.h. bestimmte Tatsachen bezüglich der Protokollausführung können nicht abgestritten werden. Das Protokoll erzeugt dann bei seiner erfolgreichen Ausführung Nachweise, die diese Tatsachen überprüfbar machen. Der Begriff *erfolgreiche Protokollausführung* bedeutet hier für eine Partei $P \in \{A, B\}$, daß P das gewünschte Objekt O_Q nach der letzten Zustandsänderung dieser Protokollausführung besitzt. Im Einzelnen gibt es bei fairem Austausch die folgenden zwei Arten von Nichtabstreitbarkeit:

Nichtabstreitbarkeit der Herkunft für P: Wenn sich P an das Protokoll hält und dessen Ausführung erfolgreich verläuft, dann kann P zweifelsfrei beweisen, daß Q der Urheber des Objekts O_Q ist.

Nichtabstreitbarkeit des Empfangs für P: Wenn sich P an das Protokoll hält und dessen Ausführung erfolgreich verläuft, dann kann P zweifelsfrei beweisen, daß Q das Objekt O_P erhalten hat oder jederzeit noch erhalten kann.

Ein Austauschprotokoll kann für jede Partei eine der folgenden vier Arten von Beweisen erzeugen:

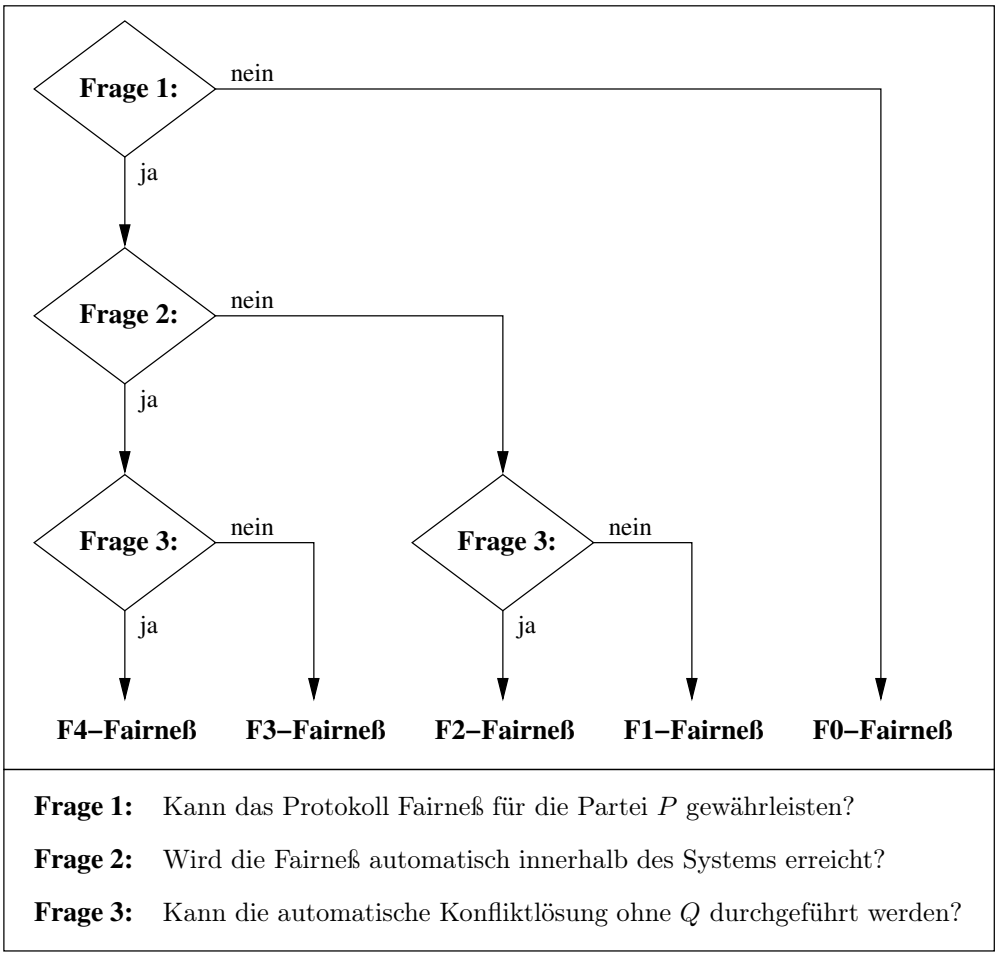

Abbildung 2.1: Der genaue Grad der Fairneß für eine Partei $P \in \{A, B\}$ kann anhand von wenigen Fragen ermittelt werden.

N3: Nichtabstreitbarkeit der Herkunft und des Empfangs

N2: Nichtabstreitbarkeit des Empfangs

N1: Nichtabstreitbarkeit der Herkunft

N0: keine nichtabstreitbaren Beweise

Wie diese nichtabstreitbaren Beweise aussehen und wie sie überprüft werden, führt zu einer weiteren Unterscheidung:

1. Eine Partei besitzt den gesamten nichtabstreitbaren Beweis und kann allein damit andere Parteien von der Gültigkeit des Beweises überzeugen.

2. Eine Partei besitzt einen Teil eines nichtabstreitbaren Beweises und kann nur mit Hilfe einer oder mehrerer weiterer Parteien andere von der Gültigkeit des Beweises überzeugen.

Obwohl einige Austauschprotokolle diese zweite Interpretation von nichtabstreitbaren Beweisen verwenden (z.B. [ASW98a] in Abschnitt 5.2 oder [FPH00, FPH01] sowie [Sch00] unter dem Begriff Drei-Parteien-Verifizierbarkeit), wird in dieser Arbeit immer die erste Variante verwendet. Dadurch wird sichergestellt, daß nicht nur ein Beweis existiert (was im 2. Fall möglich wäre), sondern daß die Partei ihn auch vollständig besitzt und jederzeit anwenden kann. Daher verwende ich auch die folgenden Bezeichnungen für die Informationen, die als Beweis dienen: Eine Partei erhält einen *Nachweis der Herkunft*, wenn das ausgeführte Protokoll Nichtabstreitbarkeit der Herkunft garantiert. Ebenso bekommt man einen *Nachweis des Empfangs*, wenn das Protokoll Nichtabstreitbarkeit des Empfangs garantiert.

Da eine der vier Arten von Beweisen N0–N3 für jede am Austausch beteiligte Person erzeugt wird, ergeben sich bei zwei Parteien insgesamt 16 verschiedene Möglichkeiten, in welchem Umfang Beweise für die Nichtabstreitbarkeit bereitgestellt werden können.

5. Optionale Protokolleigenschaften

In einigen Fällen sind die bisher definierten Protokolleigenschaften etwas zu schwach, um eine gerechte Ausführung eines Austauschprotokolls zu garantieren. Daher sollten in diesen Fällen noch manche der unter der Bezeichnung *Benachteiligungsfreiheit* zusammengefaßten optionalen Bedingungen erfüllt sein. Diese zusätzlichen Protokolleigenschaften sind im wesentlichen abhängig von den Eigenschaften der ausgetauschten Objekte. Mögliche Benachteiligungen sind im Einzelnen:

- *Verspätete Lieferung von verderblichen Objekten:* Manche Objekte wie z.B. Theaterkarten, aktuelle Börsennachrichten, etc. verlieren deutlich an Wert, wenn sie zu spät geliefert werden. Dies kann zu einer klaren Benachteiligung des Käufers führen.

- *Verlust einer auszutauschenden Zahlung:* Bei manchen Zahlungssystemen (z.B. [Jak95, But00, KV01d]) kann der Kunde sein Geld verlieren, falls während der Zahlung ein Fehler auftritt. Aus diesem Grund fordert Tygar in [Tyg96], daß Zahlungen atomar erfolgen sollen (auch *Geld-Atomizität* genannt, engl. money atomicity), um Benachteiligungen des Kunden zu vermeiden.

- *Deanonymisierung einer anonymen Partei:* Falls es sich bei einem Austausch um einen anonymen Einkauf handelt, sollte das Austauschprotokoll immer ohne Identifizierung des Kunden ausführbar sein. Nur in diesem Fall ist der Einsatz von anonymen Bezahlverfahren (wie z.B. [Cha83, Bra93, CMS96, KV01c, KV02]) sinnvoll.

- *Herstellungskosten für ein nicht ausgetauschtes Objekt:* Falls eine Partei ein gewünschtes Objekt mit nicht unerheblichem Aufwand speziell für die andere Partei herstellt, entstehen ihr bei einem fehlgeschlagenen Austausch möglicherweise trotzdem diese Herstellungskosten. Man kann daher fordern, daß die Herstellung eines solchen Objekts nur dann erfolgt, wenn der Austausch erfolgreich zu Ende geführt werden kann.

- *Verlust der Vertraulichkeit eines Objekts:* Einige Objekte wie z.B. eine E-Mail oder ein Vertrag müssen vertraulich ausgetauscht werden. In [ZDB00] wird z.B. für den Austausch einer E-Mail gegen eine Empfangsbestätigung gefordert, daß nur A und B den Inhalt der E-Mail lernen. Alle anderen Parteien, auch wenn es sich um eine vertrauenswürdige Partei wie den Vermittler handelt, sollen den Inhalt der E-Mail nicht lernen.

Diese Liste stellt nur einen kleinen Teil der möglichen Benachteiligungen dar. Eine vollständige Auflistung ist kaum möglich, da es in Abhängigkeit von den Eigenschaften der auszutauschenden Objekte immer wieder neue Formen der Benachteiligung geben kann.

2.2.4 Diskussion und Beispiele

Obwohl die vier Protokolleigenschaften Wirksamkeit, Terminierung, Faireß und Nichtabstreitbarkeit verschiedene Anforderungen an ein Protokoll für fairen Austausch stellen, gibt es zwischen der Terminierungs- und der Faireßeigenschaft eine enge Wechselwirkung. Obwohl Faireß auch für Protokolle ohne Terminierung definiert ist, macht diese Kombination beider Eigenschaften in den meisten Fällen keinen Sinn. Ein faires Protokoll endet zwar auch ohne Terminierung immer in einem fairen Zustand, aber ohne die Terminierung weiß eine Partei nicht, ob dieser faire Zustand bereits erreicht ist. Eine ähnliche Ungewißheit besteht auch bei T1-Terminierung, weil eine Partei dabei möglicherweise nicht herausfinden kann, ob sich ihr Zustand nach der Terminierung noch einmal ändern wird bzw. wann das schließlich sein wird. Aus diesen Gründen sollten Protokolle mit T0/T1-Terminierung in der Praxis immer mit einem Zeitlimit eingesetzt werden: Nach einer sehr großzügig gewählten Zeitspanne (z.B. ein Jahr, was in der Praxis allen Parteien genug Zeit für die Kommunikation geben sollte) werden Vermittler bzw. Gericht keine

Zustandsänderungen mehr veranlassen, so daß sich die Parteien doch noch Gewißheit über den Endzustand des Austausches verschaffen können, indem sie den Vermittler fragen, ob das Zeitlimit überschritten wurde. Die Implementierungsdetails für ein solches Zeitlimit werden in Abschnitt 4.2.3 diskutiert.

Die in der Regel sinnvollste Kombination von Fairneß- und Terminierungseigenschaft besteht aus F4-Fairneß und T2-Terminierung. Die T2-Terminierung stellt sicher, daß sich der Zustand nach der stets gewährleisteten Terminierung nicht mehr ändert, und aufgrund der Fairneßeigenschaft ist dieser Terminierungszustand einer Partei auch immer fair. Schließlich hat die F4-Fairneß den Vorteil, daß sie zusätzlichen Aufwand, wie ein relativ teures Gerichtsverfahren bei F1/F2-Fairneß oder eine schwer zu implementierende synchrone Kommunikationsverbindung bei F1/F3-Fairneß, vermeidet. Aus diesen Gründen werde ich mich in dieser Arbeit soweit möglich auf Protokolle konzentrieren, die Wirksamkeit, T2-Terminierung, F4-Fairneß sowie Nichtabstreitbarkeit der Herkunft und/oder des Empfangs gewährleisten können.

Nachdem das Zusammenwirken der Protokolleigenschaften genauer beleuchtet wurde, stelle ich hier die wichtigsten Fairneßdefinitionen aus der Literatur vor und ordne diese in die Fairneßhierarchie ein. Anhand dieser Beispiele zeigt sich, daß die Fairneßhierarchie andere Fairneßdefinitionen umfaßt und zudem sehr präzise Aussagen über den Fairneßgrad eines Protokolls erlaubt.

Die am weitesten verbreitete und akzeptierte Fairneßdefinition ist die *starke Fairneß* von Asokan [Aso98, ASW97a]; sie wird zum Beispiel auch in [GJM99, SM99] verwendet bzw. dient als Grundlage für im wesentlichen identische Fairneßdefinitionen (z.B. [ZDB00, FPH00, GPV99, MGK02]). Starke Fairneß umfaßt Wirksamkeit, F4/F3-Fairneß, mindestens T1-Terminierung und falls gewünscht auch N1–N3-Nichtabstreitbarkeit. Bei Austauschprotokollen mit *asynchroner Kommunikation* (z.B. [ASW98a]) entspricht starke Fairneß der F4-Fairneß, da Fairneß bei beiden Definitionen unter den gleichen Voraussetzungen realisiert wird. Einige Austauschprotokolle mit *synchroner Kommunikation* (z.B. [ASW97a, Sch00]) setzten bei starker Fairneß auch die Mitwirkung beider Parteien an der Konfliktlösung voraus, was in der Fairneßhierarchie dann F3-Fairneß entspricht. Durch die Forderung nach T1- oder T2-Terminierung ist bei starker Fairneß immer die Terminierung einer Partei sichergestellt. Die Nichtabstreitbarkeit ist nicht zwingend vorgesehen, sie wird jedoch empfohlen, um späteren Streit über den Ausgang des Austausches durch diese Beweise zu verhindern. Weitere Anforderungen beinhaltet die starke Fairneß nicht. Stattdessen interpretiert Asokan die unter optionale Protokolleigenschaften aufgeführten Anforderungen als zusätzliche Eigenschaften, die einzelne Protokolle besitzen können.

Eine andere Fairneßdefinition ist die *schwache Fairneß*, die ebenfalls von Asokan eingeführt wurde [Aso98, ASW97a]. Dabei wird entweder starke Fairneß erreicht oder eine sich korrekt verhaltende Partei erhält durch die Protokollausführung automatisch einen Beweis, daß die andere Partei das Objekt erhalten hat bzw. jederzeit erhalten kann. Dieser Beweis stellt also eine abgeschwächte Form von Nichtabstreitbarkeit des Empfangs dar. Die Idee dabei ist, daß mit einem schwach fairen Protokoll der Austausch in den meisten Fällen ohne Probleme fair durchgeführt werden kann. Falls das nicht der Fall ist, muß die benachteiligte Partei mit den erhaltenen Beweisen eine Wiederherstellung der

Fairneß außerhalb des Systems (z.B. vor Gericht) anstreben, was der F1- und F2-Fairneß entspricht.

Eine speziell auf Zahlungssysteme zugeschnittene Definition von Fairneß stammt von Tygar [Tyg96, CHTY96, Tyg98] und orientiert sich am Transaktionskonzept von Datenbanken [GR93], speziell an der Atomizität: Seine Anforderung an ein Zahlungssystem ist immer die *Geld-Atomizität* (engl. money atomicity), die meiner optionalen Eigenschaft entspricht, daß es keinen Verlust von Zahlungen geben darf. Als Alternative zum Begriff der Fairneß definiert Tygar die *Waren-Atomizität*. Diese fordert, daß die Ware genau dann geliefert wird, wenn das Geld bezahlt wird. Ein Protokoll, das Waren-Atomizität realisiert, wird also F4-Fairneß erreichen. Die Protokolleigenschaften der Wirksamkeit und Terminierung sieht Tygar implizit als immer erfüllt an. Die Nichtabstreitbarkeit der Herkunft nennt er beglaubigte Lieferung (engl. certified delivery) [Tyg96] oder auch einseitig beglaubigte Lieferung (engl. one-sided certified delivery) [CHTY96]. Die Nichtabstreitbarkeit des Empfangs nennt er beidseitig beglaubigte Lieferung (engl. two-sided certified delivery) [CHTY96].

2.3 Das Konzept eines Fairneßdienstes

In diesem Abschnitt beschreibe ich den Entwurf des Fairneßdienstes FlexiFair, dessen Komponenten in Abschnitt 2.3.1 kurz vorgestellt werden. Das Ziel dieses Fairneßdienstes besteht darin, vielen verschiedenen Anwendungen Protokolle für den fairen Austausch zur Verfügung zu stellen. Dabei konzentriere ich mich speziell auf Protokolle, die Fairneß ohne den Rückgriff auf ein teures und eventuell langwieriges Gerichtsverfahren sicherstellen können. Daher diskutiere ich zuerst in Abschnitt 2.3.2 die verschiedenen Ansätze für Fairneßprotokolle, von denen sich nur solche mit aktiv beteiligtem Vermittler und solche mit Vermittler im Fehlerfall als für einen Fairneßdienst geeignet erweisen.

Um bei der Implementierung beide Arten von Protokollen über eine einheitliche Schnittstelle ansprechen zu können, gliedere ich in Abschnitt 2.3.3 den Ablauf der Protokolle in verschiedene Module, die bei allen Protokollen die gleichen Funktionen übernehmen. Diese Modellierung von Fairneßprotokollen stellt meiner Meinung nach eine gute Abstraktion der bekannten Protokolle dar, weil

- alle in der Praxis relevanten Fairneßprotokolle damit modelliert werden können,
- alle Besonderheiten der Protokolle erhalten bleiben,
- sich die so modellierten Protokolle später leicht in verschiedene Anwendungen einbauen lassen und
- die Austauschbarkeit verschiedener Protokolle erreicht wird.

Für eine exakte Beschreibung einzelner Protokolle müssen schließlich die Eigenschaften der ausgetauschten Objekte genau definiert sein. Um die Besonderheiten einzelner Protokolle abstrakt fassen zu können, führe ich in Abschnitt 2.3.4 zusätzlich zu den aus der Literatur bekannten Begriffen der Generierbarkeit und Widerrufbarkeit [ASW97a]

die Objekteigenschaften der schwachen Generierbarkeit und schwachen Widerrufbarkeit ein. Diese ermöglichen die explizite Beschreibung von Objekteigenschaften, die in vielen Protokollen benötigt werden, aber bisher nur implizit definiert waren. Daher präzisieren diese neuen Begriffsbildungen die Modellierung von Austauschprotokollen und tragen außerdem zum besseren Verständnis der damit verbundenen Konfliktlösungsmechanismen bei, was in Kapitel 3 die Entwicklung neuer Austauschprotokolle ermöglicht.

2.3.1 Übersicht

Das Ziel eines Fairneßdienstes ist es, verschiedenen Anwendungen Fairneßprotokolle zur Verfügung zu stellen, wodurch der Einsatz von fairem Austausch deutlich erleichtert wird. Da ein Fairneßdienst Protokolle mit verschiedenen Eigenschaften unterstützen soll, muß es eine einheitliche Schnittstelle geben, damit die Besonderheiten der Protokolle vor der Anwendung verborgen sind. Daher schlage ich in Abschnitt 2.3.3 eine einheitliche Modellierung von Fairneßprotokollen vor, die eine Implementierung aller Austauschprotokolle ohne Gerichtsverfahren und ohne Drei-Parteien Konfliktlösung ermöglicht.

Wenn eine Anwendung einen Fairneßdienst zum Austausch ihrer Objekte verwenden will, dann kennt diese die Eigenschaften der ausgetauschten Objekte. Deshalb muß ein Fairneßdienst den darauf aufbauenden Anwendungen standardisierte Klassen zur Implementierung von Objekten und der Beschreibung ihrer Eigenschaften anbieten. Nur wenn eine Anwendung diese vorgefertigten Klassen verwendet, können die vom Fairneßdienst bereitgestellten generischen Austauschprotokolle die Eingaben der Anwendung verarbeiten. Daher werden in Abschnitt 2.3.4 die wesentlichen Eigenschaften von ausgetauschten Objekten diskutiert, die die Grundlage für den Entwurf von Klassen für digitale Objekte und deren Eigenschaften bilden.

Eine weitere wichtige Komponente eines Fairneßdienstes sind Klassen, die beschreiben, welches Objekt eine Partei bei einem Austausch erwartet. Damit ist die Überprüfung von empfangenen Objekten möglich, falls es eine Prüffunktion gibt, die das Objekt mit der Beschreibung vergleicht. Im folgenden wird die Existenz einer Objektbeschreibung und einer dazugehörigen Prüffunktion vorausgesetzt. Die Realisierung solcher Überprüfungen wird dann im Rahmen der Implementierung in Abschnitt 4.3 vorgestellt.

Schließlich muß ein Fairneßdienst auch einen Vermittler zur Verfügung stellen, der die Fairneß beim Austausch garantiert. Dieser Vermittler muß alle verwendeten Austauschprotokolle kennen und in der Lage sein, alle auftretenden Konflikte zu lösen.

Ein wesentliches Designziel von FlexiFair besteht darin, daß während des Betriebs am Vermittler möglichst wenige Veränderungen vorgenommen werden sollen. Insbesondere sollen Änderungen an den Anwendungen bzw. an den von diesen Anwendungen ausgetauschten Objekten keinen Einfluß auf die Arbeit des Vermittlers haben. Dieser Ansatz hat viele Vorteile:

Stabilität des Vermittlers: Da jede Änderung die Zuverlässigkeit des Vermittler verringern kann, sollten möglichst wenige Änderungen vorgenommen werden.

Höhere Vertrauenswürdigkeit des Vermittlers: Falls nur sehr selten Änderungen am Vermittler vorgenommen werden, lohnen sich eher die Kosten für eine Evaluierung

und Zertifizierung des Vermittlers. Nach einer Änderung des Vermittlers bleibt das durch die Zertifizierung bestätigte Sicherheitsniveau nicht unbedingt bestehen.

Spontaner Einsatz: Anwendungen können auf den Vermittler zugreifen, ohne daß sie sich vorher anmelden müssen oder auf eine Freischaltung von benötigten Funktionen warten müssen.

Die einzelnen Komponenten des Fairneßdienstes FlexiFair sind noch einmal in Abbildung 2.2 zusammengefaßt. Die hell dargestellten Komponenten sind Teile von FlexiFair, während die dunkleren zur Anwendung gehören. Die Objekte und deren Beschreibungen sind zwar als Klassen von FlexiFair vorgegeben, sie werden jedoch von der Anwendung entsprechend deren Anforderungen eingesetzt.

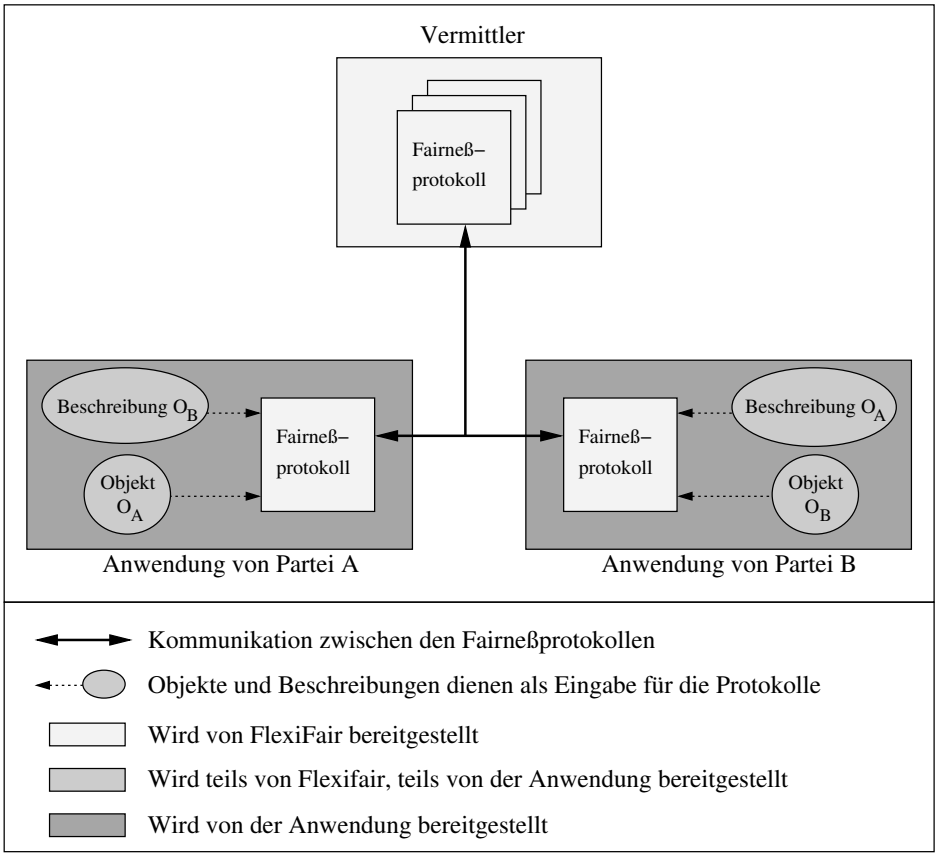

Abbildung 2.2: FlexiFair stellt Fairneßprotokolle, Klassen für die auszutauschenden Objekte und ihre Beschreibungen, sowie eine Implementierung eines Vermittlers zur Verfügung.

2.3.2 Klassifikation von Fairneßprotokollen

In der Literatur findet man eine große Anzahl von Protokollen zur Realisierung von fairem Austausch. Diese Protokolle lassen sich am besten dadurch klassifizieren, daß man sie nach der Art und Weise ordnet, in der der vertrauenswürdige Vermittler (oder auch falls nötig das Gericht als eine Erweiterung des Vermittlers) am Austausch beteiligt ist (siehe z.B. auch [Aso98]). Dies ergibt die grundlegende Einteilung in Protokolle mit aktiv beteiligtem Vermittler, mit Vermittler nur im Fehlerfall oder ohne Vermittler. Die Grundideen dieser drei Ansätze werden in diesem Abschnitt kurz zusammengefaßt und bewertet. Aus diesen Bewertungen folgt, daß Protokolle ohne Vermittler (und somit auch ohne Gericht) bei der genaueren Modellierung von Fairneßprotokollen in Abschnitt 2.3.3 nicht berücksichtigt werden müssen, da sie keine Fairneß gemäß der hier verwendeten Definition gewährleisten können. Stattdessen muß ein Fairneßdienst die Protokolle mit aktivem Vermittler und mit Vermittler im Fehlerfall unterstützen.

Protokolle mit aktivem Vermittler

Die einfachsten Protokolle für fairen Austausch verwenden einen *aktiven Vermittler* (siehe z.B. [Rab83, BP90, CHTY96, ZG96, FR97]). Dabei schicken beide Parteien ihre Objekte dem Vermittler, der diese überprüft und schließlich an die jeweils andere Partei weiterleitet. Die Fairneß ist bei diesen Protokollen auf einfache Weise erfüllt, da der Vermittler den Austausch der Objekte nur genau dann vollendet, wenn deren Überprüfung erfolgreich verlaufen ist. Dadurch ist sichergestellt, daß entweder jede Partei das gewünschte Objekt erhält oder daß keine Partei etwas bekommt.

Vorteile: Der Vermittler weiß immer genau über den Zustand des Austausches bescheid und kann daher immer Fairneß garantieren. Außerdem werden keine besonderen Eigenschaften von den ausgetauschten Objekten gefordert.

Nachteile: Der Vermittler kann zum Engpaß werden, wenn er gleichzeitig viele Austauschtransaktionen durchführen muß. Ohne ihn kann jedoch kein fairer Austausch durchgeführt werden, so daß schon ein kurzzeitiger Ausfall des Vermittler schwere Probleme verursachen kann, falls kein alternativer Vermittler zur Verfügung steht.

Protokolle mit Vermittler nur im Fehlerfall

Diese Art von Protokollen mit Beteiligung des Vermittlers nur im Fehlerfall (auch *optimistische Protokolle* genannt [ASW97a]) ist durch die Annahme motiviert, daß bei den meisten Austauschvorgängen kein Fehler auftritt bzw. keine Partei einen Betrugsversuch unternimmt. Daher wird eigentlich kein vertrauenswürdiger Vermittler benötigt. Stattdessen werden bei optimistischen Protokollen lediglich Hinweise bezüglich des Protokollablaufs gesammelt, damit bei einem Fehler der Vermittler später in der Lage ist, den tatsächlichen Ablauf des Austausches nachzuvollziehen und eine Konfliktlösung anzubieten.

Falls der Vermittler eine automatische Konfliktlösung durchführen soll, müssen die ausgetauschten Objekte jedoch die zusätzliche Eigenschaft der Generierbarkeit oder Widerrufbarkeit durch den Vermittler besitzen [Aso98, ASW97a]. Diese Objekteigenschaften werden in Abschnitt 2.3.4 definiert und anhand von Beispielen erläutert. Die genauen Voraussetzungen für optimistische Protokolle werden schließlich in Kapitel 3 untersucht.

Vorteile: Solange keine Fehler auftreten, wird kein Vermittler benötigt, so daß optimistische Protokolle sehr gut skalieren.

Nachteile: Falls keines der ausgetauschten Objekte generierbar oder widerrufbar ist, können optimistische Protokolle keine Fairneß erreichen (mehr dazu siehe Kapitel 3).

Protokolle ohne Vermittler

Die Beteiligung eines vertrauenswürdigen Dritten stellt einen zusätzlichen Aufwand dar, den man natürlich vermeiden möchte. In [EY80], sowie in [PG99] wird jedoch gezeigt, daß sich Fairneß ohne eine dritte Partei (also ein Vermittler oder ein Gericht) nicht erreichen läßt. Trotzdem gibt es eine Reihe von Protokollen, die den Austausch ganz ohne Vermittler durchführen.

Viele Protokolle verwenden das Konzept des *schrittweisen Austausches* (engl. gradual exchange), bei dem ein Austausch durch eine Vielzahl von abwechselnd gesendeten Nachrichten realisiert wird. Bei einigen dieser Protokolle (z.B. [Rab83, BOGMR90, MR99]), die auch als probabilistisch bezeichnet werden, führt eine Unterbrechung der Kommunikation zum richtig geratenen Zeitpunkt dazu, daß eine Partei das gewünschte Objekt erhält, während die andere nichts bekommt. Andere Protokolle mit schrittweisem Austausch (z.B. [Eve81, EGL82, Blu83, Ted85, BCDvdG87, Dam93, BT94, BN00]) versuchen den Vorteil durch eine Kommunikationsunterbrechung möglichst gering zu halten. Falls ein solcher Austausch unterbrochen wird, endet dieser jedoch in einem undefinierten Zustand, da keine Partei wissen kann, ob die andere ausreichendes Wissen über das Objekt erhalten hat oder nicht. Wenn zum Beispiel 30 Prozent einer digitalen Signatur übertragen wurden, dann ist unklar, ob der Empfänger daraus die vollständige Signatur berechnen kann.

Eine andere Variante des schrittweisen Austausches [SL95, San97, ZL99] zerlegt eine Austauschtransaktion in mehrere Schritte, in denen jeweils kleine Teile der Objekte ausgetauscht werden. Ein Beispiel ist der Austausch einer Musik-CD gegen Geld, der schrittweise durch das Tauschen eines Liedes gegen den entsprechenden Anteil des Geldes realisiert werden kann. Der Austausch eines Teils wird dann allerdings unfair durchgeführt, so daß eine Partei dieses Risiko akzeptieren muß. Ein weiterer Nachteil besteht darin, daß viele Objekte (z.B. Verträge) nicht in diesem Sinne teilbar sind.

Ein anderer Ansatz für einen Austausch ohne Vermittler ist der *rationale Austausch* [Jak95, Syv98, BH01]. Dieser kann nur verhindern, daß eine Partei einen Vorteil erzielt. Die Fairneß wird dabei jedoch verletzt, weil beide Parteien etwas hergeben müssen und nur eine Partei etwas erhalten kann, während die andere eventuell nichts bekommt.

Allen Ideen ohne vertrauenswürdigen Dritten besitzen jedoch die Gemeinsamkeit, daß sie gemäß der Fairneßdefinition aus Abschnitt 2.2.3 keine Fairneß sicherstellen können.

Außerdem ist die Praxisrelevanz dieser Protokolle dadurch eingeschränkt, daß sie sehr spezielle Objekteigenschaften für den Austausch fordern.

Vorteile: Der Aufwand und die Kosten für den Vermittler entfallen bei diesen Protokollen. Der Austausch kann spontan gestartet werden, da man sich nicht auf einen von beiden Parteien akzeptierten Vermittler einigen muß.

Nachteile: Für die meisten Anwendungen ist der von diesen Protokollen gebotene Grad der Fairneß zu gering bzw. die Effizienz nicht ausreichend. Außerdem sind diese Protokolle nur für Objekte mit speziellen Eigenschaften verfügbar.

2.3.3 Modellierung von Fairneßprotokollen

In diesem Abschnitt stelle ich eine auf der Veröffentlichung [VPG99] basierende abstrakte Modellierung von Fairneßprotokollen vor, die sowohl Protokolle mit aktivem Vermittler als auch solche mit Beteiligung des Vermittlers im Fehlerfall umfaßt. Dabei unterteile ich die Austauschprotokolle grundsätzlich in die folgenden 5 *Module*, die gemäß ihrer Funktion benannt sind:

M1: Aushandeln

M2: Vorbereiten des Austausches

M3: Austausch

M4: Konfliktlösung durch Fortsetzen des Austausches

M5: Konfliktlösung durch Zurücksetzen des Austausches

Jedes Austauschprotokoll muß mindestens eine Implementierung der ersten drei Module besitzen, während die Konfliktlösungsmodule optional sind. Falls auf diese automatische Konfliktlösung durch den Vermittler verzichtet wird, kann jedoch höchstens F1/F2-Fairneß mit dem Gericht erreicht werden. Im folgenden werden daher speziell Protokolle mit Konfliktlösung betrachtet, um F4-Fairneß gewährleisten zu können.

Bei der folgenden Beschreibung der einzelnen Module bezeichne ich mit der Partei $P \in \{A, B\}$ immer die Partei, die das Modul gerade ausführt, während Q der Bezeichner für die jeweils andere Partei ist (also $Q \in \{A, B\} \setminus \{P\}$). Von den 5 Modulen müssen die ersten drei jeweils von beiden Parteien ausgeführt werden, um den Austausch erfolgreich durchzuführen. Ein Konfliktlösungsmodul wird möglicherweise nur von einer Partei P ausgeführt, während die andere Partei Q dieses Modul nicht benötigt, da aus ihrer Sicht der Austausch fehlerfrei verlaufen sein kann.

Modul M1: Aushandeln

Vorbedingung: Zwei Parteien beabsichtigen einen fairen Austausch durchzuführen.

Beteiligte Parteien: P und Q

Funktion: Die Parteien P und Q handeln die Details aus, bevor der faire Austausch gestartet werden kann. Dabei müssen sie insbesondere die folgenden Fragen klären:

- Welches Protokoll soll verwendet werden?
- Welcher Vermittler V soll verwendet werden?
- Welche Objekte erwarten P und Q?
- Welche Partei beginnt mit der Ausführung des Protokolls?
- Welche Signaturschlüssel verwenden die Parteien, um die Authentizität von Nachrichten zu gewährleisten?

Das Aushandeln dieser Protokollparameter muß immer vor dem Start des eigentlichen Austauschprotokolls erfolgen, so daß dieses Modul entweder Teil der Anwendung ist, die fairen Austausch durchführen will, oder durch ein allgemein verwendbares Aushandlungsprotokoll realisiert werden muß.

Fehlerbehandlung: Sollte beim Aushandeln ein Fehler auftreten, wird der Austausch erst gar nicht gestartet. Da niemand sein Objekt weitergegeben hat, ist die Fairneßeigenschaft trivialerweise erfüllt.

Ergebnis: P und Q haben sich auf die Einzelheiten der Durchführung des Austausches geeinigt.

Modul M2: Vorbereiten des Austausches

Vorbedingung: M1 wurde ausgeführt.

Beteiligte Parteien: P, sowie Q oder V

Funktion: In diesem Modul führt P alle Aktionen aus, die vor dem eigentlichen Austausch der Objekte notwendig sind. Dazu gehört insbesondere die Überprüfung von bestimmten Objekteigenschaften, die in Abschnitt 2.3.4 ausführlich diskutiert werden. In diesem Modul wird auch sichergestellt, daß eventuell nötige Konfliktlösungen mittels Modul M4 oder M5 ausreichend vorbereitet sind.

Fehlerbehandlung: Da in diesem Modul noch keine Objekte ausgetauscht werden, wird bei einem Fehler meist das Zurücksetzen des Austausches angestrebt. Falls P noch keine relevanten Informationen an Q gesendet hat, kann P den Austausch einfach beenden. Ansonsten muß er den Abbruch des Austausches mittels M5 durchführen. Als Alternative steht eventuell auch die Fortsetzung des Austausches mittels M4 zur Verfügung.

Ergebnis: Der Austausch kann danach gestartet werden.

Modul M3: Austausch

Vorbedingung: M2 wurde ausgeführt.

Beteiligte Parteien: P, sowie Q oder V

Funktion: In diesem Modul wird der Austausch der Objekte durchgeführt. Falls kein Fehler auftritt hat nach Beendigung dieses Moduls die Partei P das gewünschte Objekt erhalten und der Austausch ist beendet.

Fehlerbehandlung: Falls ein Fehler auftritt muß P auf jeden Fall eine Konfliktlösung initiieren. Das Ziel dieser Konfliktlösung kann entweder das erfolgreiche Beenden mittels Modul M4 oder das Zurücksetzen des Austausches mittels Modul M5 sein, was beides die Fairneß sicherstellen würde. Falls der Partei P beide Optionen zur Konfliktlösung zur Verfügung stehen, sollte P immer zuerst Modul M4 ausführen, da eine Fortsetzung des Austausches eher im Interesse beider Parteien ist als das Verhindern des Austausches durch das Zurücksetzen mittels Modul M5.

Ergebnis: P hat das Objekt O_Q erhalten und beendet den Austausch.

Modul M4: Konfliktlösung durch Fortsetzen des Austausches

Vorbedingung: Die Möglichkeit zur Konfliktlösung wurde in M2 vorbereitet und ein Fehler ist aufgetreten.

Beteiligte Parteien: P und V (die Beteiligung von Q an der Konfliktlösung ist nur unter den Vorbedingungen für F3/F1-Fairneß möglich)

Funktion: P wendet sich an den Vermittler, damit sich dieser für eine erfolgreiche Beendigung des Austausches einsetzt. Falls der Vermittler erfolgreich ist, wird P das gewünschte Objekt O_Q erhalten.

Fehlerbehandlung: Falls der Vermittler nicht in der Lage ist, das Protokoll fortzusetzen, kann es mehrere Gründe dafür geben: Falls das Protokoll bereits zurückgesetzt wurde, wird der Vermittler P über diese Entscheidung informieren. Falls der Vermittler aufgrund fehlerhafter Daten von P den Austausch nicht fortsetzen kann, wird er keine Entscheidung treffen. P muß dann entweder Modul M5 ausführen oder M4 mit den richtigen Daten nochmal ausführen.

Ergebnis: P hat das Objekt O_Q erhalten und beendet den Austausch.

Modul M5: Konfliktlösung durch Zurücksetzen des Austausches

Vorbedingung: Die Möglichkeit zur Konfliktlösung wurde in M2 vorbereitet und ein Fehler ist aufgetreten.

Beteiligte Parteien: P und V (die Beteiligung von Q an der Konfliktlösung ist nur unter den Vorbedingungen für F3/F1-Fairneß möglich)

Funktion: P wendet sich an den Vermittler, damit dieser den unvollständigen Austausch zurücksetzt. Falls der Vermittler erfolgreich ist, besitzt weder P das Objekt O_Q, noch Q das Objekt O_P.

Fehlerbehandlung: Falls der Vermittler nicht in der Lage ist, den Austausch zurückzusetzen, kann es zwei Gründe dafür geben: Einerseits kann der Vermittler den Austausch bereits fortgesetzt haben. Dann kann er P jedoch immer das gewünschte O_Q liefern. Andererseits können die von P gesendeten Daten fehlerhaft oder unvollständig sein, so daß der Vermittler keine Konfliktlösung durchführen kann. Dann sollte P entweder M5 nochmal mit korrekten Daten aufrufen oder M4 ausführen.

Ergebnis: Keine Partei besitzt das Objekt der anderen Partei, und P beendet den Austausch.

Diskussion dieser Modellierung

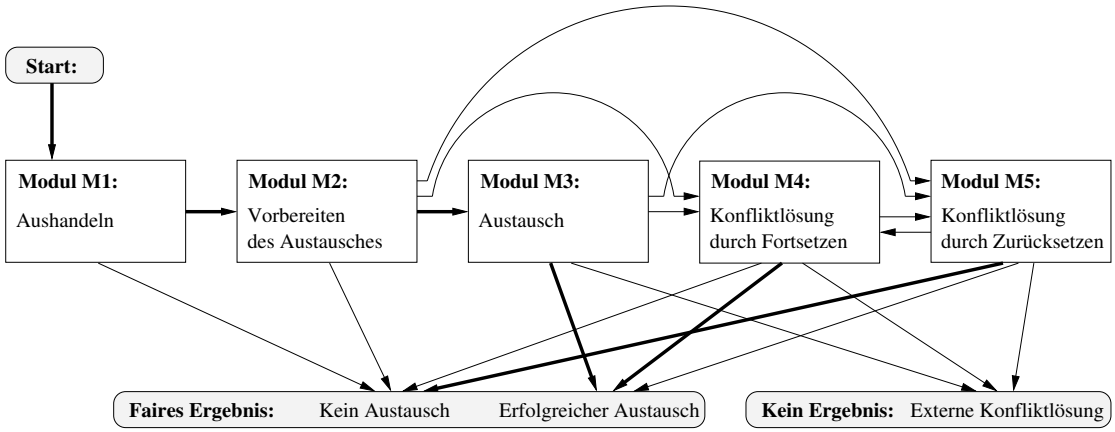

Abbildung 2.3: Modulare Zerlegung von Fairneßprotokollen. Dicke Pfeile zeigen das Ergebnis einer erfolgreichen Modulausführung an.

Die fünf Module eines Fairneßprotokolls können in der in Abbildung 2.3 dargestellten Reihenfolge abgearbeitet werden. Alle Protokolle mit F4/F3-Fairneß beenden den Austausch entweder erfolgreich, was mit $\langle$*erfolgreicher Austausch*$\rangle$ bezeichnet wird, oder keine Partei erhält etwas, so daß das Ergebnis $\langle$*kein Austausch*$\rangle$ lautet. Der Vollständigkeit halber ist in Abbildung 2.3 auch die Möglichkeit der externen Konfliktlösung mittels Gericht dargestellt, die bei Protokollen mit F1/F2-Fairneß als ein weiteres mögliches Ergebnis auftreten kann. Da der auf dieser Modellierung basierende Fairneßdienst FlexiFair nur Protokolle ohne Gerichtsverfahren unterstützen soll, wird die Möglichkeit einer externen Konfliktlösung im folgenden größtenteils nicht mehr betrachtet.

Ein fehlerfreier Austausch wird immer durch die Abfolge der Module (1,2,3+$\langle$*erfolgreicher Austausch*$\rangle$) beschrieben, da das Endergebnis $\langle$*erfolgreicher Aus-*

tausch⟩ erreicht wird, ohne auf eine Konfliktlösung mittels M4 oder M5 zurückzugreifen. In Abbildung 2.3 ist dieser fehlerfreie Fall auch durch die dicken Pfeile besonders hervorgehoben, die das Ergebnis einer fehlerfreien Modulausführung anzeigen.

Falls Fehler auftreten, ergibt sich in der Regel die Ausführung der Modulfolgen (1,2,3,4+ ⟨*erfolgreicher Austausch*⟩), (1,2,3,4,5+⟨*kein Austausch*⟩), (1,2,4+⟨*erfolgreicher Austausch*⟩), (1,2,4,5+⟨*kein Austausch*⟩), (1,2,5+⟨*kein Austausch*⟩), (1,2,5,4+⟨*erfolgreicher Austausch*⟩) oder (1,2,3,5+⟨*kein Austausch*⟩). Diese Art der Notation sagt aus, daß in dem Modul vor den Konfliktlösungsmodulen M4 und M5 ein Fehler aufgetreten ist, der durch M4 bzw. M5 behoben werden soll. Falls sowohl M4 als auch M5 ausgeführt werden, so ist der erste Versuch der Konfliktlösung fehlgeschlagen. Die Ausführungsreihenfolge (1,2,3,4,5+⟨*kein Austausch*⟩) entsteht zum Beispiel, wenn in Modul M3 ein Fehler auftritt, die Fortsetzung des Austausches durch den Vermittler in M4 scheitert und schließlich der Austausch durch den Vermittler in M5 erfolgreich zurückgesetzt wird. Schließlich wird die Ausführung dieses Beispiels mit dem Ergebnis ⟨*kein Austausch*⟩ beendet.

Die Ausführungsreihenfolge (1,2,3,5,4+⟨*erfolgreicher Austausch*⟩) ist zwar prinzipiell auch möglich, sie sollte jedoch nicht verwendet werden: Da Modul M2 erfolgreich ausgeführt wurde, ist auch die Möglichkeit zur Fortsetzung des Austausches in Modul M4 erfolgreich vorbereitet worden. Dann sollte statt dem Zurücksetzen des Austausches in Modul M5 immer erst das Fortsetzen mit Modul M4 versucht werden, weil dies direkt zu einer erfolgreichen Beendigung des Austausches führen kann. Erst als letztes Mittel sollte schließlich das Zurücksetzen des Austausches mit Modul M5 veranlaßt werden.

Diese Vielzahl der möglichen Protokollabläufe zeigt, wie flexibel diese Modellierung ist. Sie erlaubt auch verschiedene Ausführungsreihenfolgen bei der Konfliktlösung, so daß eine Anwendung auf einen Fehler bei der Protokollausführung angemessen reagieren kann.

2.3.4 Eigenschaften der ausgetauschten Objekte

In dieser Arbeit werden nur *digitale Objekte* betrachtet (engl. digital items). Diese lassen sich jeweils durch einen endlichen Bitstring darstellen bzw. abspeichern und können durch den Versand einer Nachricht über ein Kommunikationsmedium an andere Parteien transferiert werden. Alle digitalen Objekte müssen die Eigenschaft der *Idempotenz* erfüllen: Wird dasselbe Objekt mehrmals an eine Partei geschickt, so bedeutet der mehrfache Empfang dieses Objekts keinen Unterschied zum einfachen Empfang des Objekts.

Eine Verallgemeinerung von digitalen Objekten sind *transferierbare Objekte* (engl. „forwardable items" [Aso98] oder „transferable items" [Sch00, LPSW00]). Im Gegensatz zu digitalen Objekten müssen diese nicht in eine einzelne Nachricht kodierbar sein, sondern sie werden mit einem speziellen Transferprotokoll übertragen. Beispiele für solche durch ein interaktives Protokoll transferierbaren Objekte sind Multimediadaten oder Software mit interaktiv berechneten Kopierschutzmerkmalen (z.B. [PS96, PW97]) oder elektronische Zahlungen (z.B. [KV01d]).

Obwohl Protokolle zum fairen Austausch von transferierbaren Objekten vorgeschlagen wurden [Aso98, ASW98a, Sch00], zeigt eine genauere Betrachtung, daß diese derart mo-

dellierten Objekte nicht generell als Grundlage für eine Implementierung dienen können: Da der Vermittler spätestens im Fehlerfall die Rolle einer Partei beim Transferprotokoll allein übernehmen muß, scheitert er bei der Durchführung des Transferprotokolls, wenn die am Transfer beteiligten Parteien über spezielle Geheimnisse verfügen müssen, so wie es z.B. bei Kopierschutzmerkmalen [PS96, PW97] der Fall ist. Es lassen sich also nicht alle transferierbaren Objekte mit einem generischen Fairneßdienst austauschen. Außerdem müssen auch transferierbare Objekte während der Konfliktlösung vom Vermittler speicherbar sein. Für das Speichern muß aber immer eine Darstellung als Bitstring existieren, so daß die *austauschbaren* transferierbaren Objekte äquivalent zu digitalen Objekten sind. Daher beschränke ich mich bei den ausgetauschten Objekten auf digitale Objekte, die in jedem Fall problemlos austauschbar sind.

Bei optimistischen Protokollen müssen die Objekte noch weitere Eigenschaften besitzen, damit der Austausch fair durchgeführt werden kann. Es handelt sich dabei um die Generierbarkeit bzw. Widerrufbarkeit von Objekten [ASW97a].

Generierbarkeit

Ein generierbares Objekt besitzt die Eigenschaft, daß es vom Vermittler auf Anfrage bereitgestellt werden kann. Dabei schlage ich vor, zwei unterschiedlich starke Ausprägungen von Generierbarkeit zu unterscheiden:

Starke Generierbarkeit: Ein stark generierbares Objekt kann der Vermittler garantiert auf Anfrage bereitstellen.

Schwache Generierbarkeit: Bei einem schwach generierbaren Objekt wird der Vermittler versuchen, das Objekt auf Anfrage bereitzustellen, aber er wird möglicherweise dabei scheitern. In diesem Fall kann er jedoch immer feststellen, welche Partei dafür verantwortlich ist.

Die starke Generierbarkeit (im folgenden auch einfach nur Generierbarkeit genannt [ASW97a]) eines Objekts ermöglicht den Einsatz von Protokollen, die einen Vermittler nur im Fehlerfall benötigen. Wenn eine Partei das generierbare Objekt während des Austausches nicht erhalten hat, kann sie es immer noch vom Vermittler bei der Konfliktlösung erhalten.

Da die starke Generierbarkeit oft nur mit großem Aufwand erreicht werden kann, habe ich zusätzlich den Begriff der schwachen Generierbarkeit definiert. Dadurch wird ein Anreiz geschaffen, sich korrekt zu verhalten, weil jeder Betrug zum Verhindern der Generierbarkeit erkannt wird. Deshalb sollte in den meisten Fällen das Generieren des Objekts erfolgreich sein. Aber selbst wenn das Generieren scheitern sollte, erleichtert das Erkennen der betrügerischen Partei mittels schwacher Generierbarkeit die Konfliktlösung durch den Vermittler: Da Fairneß per Definition nur für sich korrekt verhaltende Parteien garantiert werden muß, kann der Vermittler jetzt auch Maßnahmen ergreifen, die die Partei benachteiligen, die nachweislich betrogen hat.

Der Grundgedanke bei Generierbarkeit besteht darin, eine Lösung für den Fall von Kommunikationsproblemen zwischen den Parteien bereitzustellen. Zusätzlich existiert

noch das Problem des mangelnden Vertrauens unter den Parteien: Bei schwacher Generierbarkeit muß eine Partei auf das korrekte Verhalten der anderen Partei vertrauen. Es gibt allerdings einen starken Anreiz, sich korrekt zu verhalten, da ein Betrug immer erkannt wird. Dagegen ist bei starker Generierbarkeit kein Vertrauen in eine Partei außer dem Vermittler notwendig.

Um die Generierbarkeit eines Objekts schon während der Vorbereitung des Austausches in Modul M2 überprüfen zu können, wird der Besitzer des Objekts einen Beweis für dessen Generierbarkeit durch den Vermittler liefern. Nur wenn dieser *Generierungsinformation* genannte Beweis korrekt ist, ist der Empfänger von der schwachen bzw. starken Generierbarkeit überzeugt. Außerdem wird die Generierungsinformation üblicherweise noch Informationen enthalten, die der Vermittler zum Generieren des Objektes benötigt.

Beispiele. Die folgenden Beispiele illustrieren die beiden Definitionen von Generierbarkeit: Eine Partei kann ihr Objekt generierbar machen, indem sie es vor dem Austausch an den Vermittler schickt. Dieser überprüft, ob es einer gegebenen Beschreibung entspricht und speichert es. Dann stellt der Vermittler der Partei eine Bescheinigung aus, mit der diese gegenüber anderen Parteien nachweisen kann, daß der Vermittler das Objekt auf Anfrage liefern kann.

Eine andere Methode besteht darin, daß eine Partei A ihr Objekt O_A mit einem zufälligen symmetrischen Schlüssel r verschlüsselt und diesen Schlüssel daraufhin mit dem öffentlichen Schlüssel des Vermittlers V verschlüsselt. Diese beiden Chiffretexte $e_r(O_A)$ und $E_V(r)$ zusammen mit der darüber erstellten Signatur $Sig_A(e_r(O_A), E_V(r))$ dienen einer anderen Partei B als Nachweis der schwachen Generierbarkeit. Die Generierungsinformation für O_A besteht also aus den folgenden Werten:

$$geninfo_{O_A} = (e_r(O_A),\ E_V(r),\ Sig_A(e_r(O_A), E_V(r)))$$

Der Vermittler kann anhand dieser Information den zufälligen symmetrischen Schlüssel r entschlüsseln und prüfen, ob sich damit das Objekt O_A entschlüsseln läßt. Falls das Generieren dieses Objekts fehlschlägt, weiß der Vermittler bei einer gültigen Signatur von A, daß dieser sein Objekt nicht korrekt verschlüsselt und so die Generierbarkeit verhindert hat. Mit dieser Methode kann also *jedes* beliebige digitale Objekt schwach generierbar gemacht werden.

Neben diesen Beispielen für die Generierbarkeit beliebiger Objekte können auch spezielle Objekteigenschaften Einfluß auf das Generieren haben. Für digitale Signaturen wurden verschiedene besonders effiziente Methoden entwickelt, um diese stark generierbar zu machen. Diese Verfahren werden oft unter dem Begriff *nachprüfbare Hinterlegung* (engl. verifiable escrow) von Signaturen zusammengefaßt [Mao97, ASW00]: Eine Idee zur Realisierung [BF98, Che98] nutzt *konvertierbare Signaturen* [BCDP90]. Dabei handelt es sich um nichtabstreitbare Signaturen [CvA90, Cha90], die der Vermittler in gewöhnliche, von jedem überprüfbare Signaturen umwandeln kann. Der Empfänger der konvertierbaren Signatur prüft deren Generierbarkeit, indem er sich beweisen läßt, daß es eine korrekte nichtabstreitbare Signatur ist. Dadurch kann die Partei sicher sein, daß dem Vermittler die Umwandlung in eine von jedem überprüfbare Signatur gelingen wird, wodurch starke Generierbarkeit garantiert wird.

Eine andere Idee ist die *nachprüfbare Verschlüsselung* (engl. verifiable encryption) digitaler Signaturen [Mao97, ASW98b, BDM98, CD98, Ate99]. Dabei erhält eine Partei eine für den Vermittler verschlüsselte digitale Signatur. Zusätzlich wird bewiesen, daß es sich bei dem Chiffretext wirklich um die gewünschte Signatur handelt. Daher kann der Vermittler immer die Signatur entschlüsseln, was starke Generierbarkeit sicherstellt.

Eine Variante der nachprüfbaren Verschlüsselung digitaler Signaturen verwendet im voraus vom Vermittler ausgestellte Kupons [ASW97c, WV01]: Nachdem eine Partei eine gewisse Anzahl solcher Kupons vom Vermittler erhalten hat, kann sie damit später ihre Signaturen nachprüfbar verschlüsseln. Durch die vom Vermittler signierten Kupons vereinfacht sich dann das Überprüfen der Verschlüsselung deutlich.

Die mit diesen verschiedenen Methoden erreichte starke Generierbarkeit von digitalen Signaturen eignet sich insbesondere für den Austausch bei einer Vertragsunterzeichnung. In diesem Spezialfall ist aber auch eine andere, besonders einfache Realisierung der Generierbarkeit möglich: Wenn eine Partei dem Vermittler die Vollmacht erteilt, unter bestimmten Bedingungen Signaturen in seinem Namen auszustellen, kann der Vermittler einen Ersatz für eine fehlende Signatur dieser Partei liefern. Auf diese Weise kann der Vermittler einen gültigen Vertrag generieren, wenn die vom Vermittler gelieferte Ersatzsignatur als gleichwertig anerkannt wird.

Bei einer elektronischen Zahlung sind die eben beschriebenen Methoden für die starke Generierbarkeit einer Signatur eventuell nicht hinreichend für starke Generierbarkeit der Zahlung [Vog03]: Bei *digitalen Münzen* (z.B. [Cha83, Sch97]) besteht eine Zahlung im Prinzip aus einer Signatur der Bank, die diese Münzen herausgibt. Da eine solche Münze aber nur einmal zum Bezahlen verwendet werden kann, ist sie nur dann gültig, wenn sie zum ersten Mal bei der Bank eingereicht wird. Wenn eine Partei z.B. durch nachprüfbare Verschlüsselung die starke Generierbarkeit der Münzsignatur nachgewiesen hat, so kann die Zahlung mit dieser Münze also immer noch scheitern, wenn diese Münze inzwischen für eine andere Zahlung verwendet wurde. Da in diesem Fall die Bank bestätigen kann, daß die Bezahlung wegen mehrfachen Ausgebens der Münze gescheitert ist, kann man bei diesem Einsatz von überprüfbarer Verschlüsselung immerhin noch schwache Generierbarkeit der Zahlung erreichen.

Widerrufbarkeit

Ein widerrufbares Objekt besitzt die Eigenschaft, daß es vom Vermittler widerrufen und damit für den Empfänger wertlos gemacht werden kann. Auch hier schlage ich vor, zwei unterschiedlich starke Ausprägungen dieser Eigenschaft zu unterscheiden:

Starke Widerrufbarkeit: Ein stark widerrufbares Objekt kann der Vermittler garantiert für den Empfänger wertlos machen.

Schwache Widerrufbarkeit: Bei einem schwach widerrufbaren Objekt wird der Vermittler versuchen, das Objekt für den Empfänger wertlos zu machen, aber der Vermittler wird möglicherweise scheitern. In diesem Fall kann er jedoch feststellen, daß der Empfänger wirklich das gewünschte Objekt erhalten hat bzw. jederzeit erhalten kann.

Die starke Widerrufbarkeit (im folgenden auch einfach nur Widerrufbarkeit genannt) eines Objekts ermöglicht den Einsatz von Protokollen, die einen Vermittler nur im Fehlerfall benötigen. Wenn eine Partei A nachweislich beim Austausch das Objekt O_B nicht erhalten hat, kann der Vermittler das bereits an B geschickte Objekt O_A widerrufen.

Beispiele. Eine Motivation für die Definition von schwacher Widerrufbarkeit liefert das folgende Beispiel: Angenommen eine Zahlung mit einem elektronischen Zahlungssystem kann vom Vermittler bei der Bank widerrufen werden, was starker Widerrufbarkeit entsprechen würde. Falls der Widerruf erst nach einer längeren Verzögerung geschieht, also z.B. nach zwei Wochen, so hat der Empfänger möglicherweise bereits das ganze Geld abgehoben und sein Konto aufgelöst. Ein Zurückbuchen des Geldes ist dann nicht mehr möglich, aber stattdessen wird der Vermittler von der Bank einen Nachweis erhalten, daß diese Person das Geld wirklich erhalten hat.

Eine andere Implementierung von schwacher Widerrufbarkeit wurde in [ASW97a, Abschitt 3.6] vorgeschlagen: Eine Partei kann ihr Objekt beim Vermittler hinterlegen und vereinbart einen Termin, an dem der Vermittler das Objekt veröffentlicht. Bis zu diesem Termin hat die Partei noch die Möglichkeit die Weitergabe ihres Objekts zu verhindern, indem sie den Vermittler überzeugt, daß sie das Objekt der anderen Partei nicht erhalten hat. Nachdem der Veröffentlichungstermin erreicht ist, hat die andere Partei beweisbar Zugriff auf das hinterlegte Objekt, weshalb schwache Widerrufbarkeit sichergestellt ist.

2.4 Zusammenfassung

In diesem Kapitel habe ich zuerst motiviert, warum ein generischer Fairneßdienst wie FlexiFair deutliche Vorteile gegenüber nicht flexibel einsetz- und erweiterbaren Lösungen hat: Viele Anwendungen können die von FlexiFair bereitgestellten Protokolle nutzen, wodurch sich Kosten sparen lassen und die Sicherheit erhöht werden kann.

Mit der in Abschnitt 2.2 vorgeschlagenen Fairneßhierarchie lassen sich die Eigenschaften von Fairneßprotokollen sehr detailliert spezifizieren, was den Vergleich und die Auswahl von Fairneßprotokollen deutlich vereinfacht. Die im folgenden Kapitel vorgestellten Protokolle greifen dann auf diese Definition der Fairneßhierarchie zurück. Dadurch ist eine im Vergleich zu bisher in der Literatur verwendeten Definitionen präzisere Beschreibung der Protokolleigenschaften möglich.

Der Fairneßdienst FlexiFair besteht im wesentlichen aus der Implementierung von Fairneßprotokollen, den vorgefertigten Klassen für Objekte und deren Beschreibungen sowie dem Vermittler. Damit Anwendungen über die gleiche Schnittstelle auf verschiedene Austauschprotokolle zugreifen können, habe ich in Abschnitt 2.3.3 eine einheitliche Modellierung von Protokollen mit aktivem Vermittler und optimistischen Protokollen vorgeschlagen. Dadurch kann ein Fairneßprotokoll ohne Änderung der Anwendung gegen ein anderes Protokoll ausgewechselt werden.

FlexiFair beschränkt sich auf den Austausch digitaler Objekte, die sich immer mittels einer einzigen Nachricht austauschen lassen, weil nur dann die korrekte Behandlung der ausgetauschten Objekte möglich ist. Als zusätzliche Eigenschaften können die Objekte

starke und schwache Generierbarkeit bzw. Widerrufbarkeit besitzen. Der Nutzen dieser von mir vorgeschlagenen Begriffe wird im folgenden Kapitel sichtbar, da diese Begriffe die Voraussetzung für neue Fairneßprotokolle sind.

3 Modulare Austauschprotokolle

In diesem Kapitel zeige ich, wie sich verschiedene Arten von Fairneßprotokollen mit der in Abschnitt 2.3.3 eingeführten Modellierung darstellen lassen. In Abschnitt 3.1 stelle ich die generischen Realisierungen aller in der Praxis relevanten Fairneßprotokolle vor, die ohne Gerichtsverfahren und ohne synchrone Kommunikation auskommen. Zuerst beschreibe ich in Abschnitt 3.1.2 ein Protokoll mit aktivem Vermittler und mögliche Variationen. Anschließend werden systematisch verschiedene optimistische Protokolle vorgestellt, die für ihren Einsatz unterschiedliche Objekteigenschaften voraussetzen. Durch diese systematische Untersuchung der optimistischen Protokolle gelingt mir die Entwicklung der neuen Austauschprotokolle P3 und P4 (siehe auch [Vog03]), die speziell die Widerrufbarkeit von Objekten ausnutzen und auf diese Weise F4-Fairneß und T2-Terminierung garantieren. Alle bisher bekannten optimistischen Protokolle mit vergleichbaren Eigenschaften (z.B. [ASW98a, GJM99, ZDB99, MS01]) verwenden ein auf Generierbarkeit basierendes Konstruktionsprinzip, so daß die von mir entwickelten Protokolle neuartige Konstruktionsprinzipien beschreiben.

Ein weiteres Ergebnis der systematischen Untersuchung von optimistischen Protokollen zeigt sich bei der Betrachtung der von mir eingeführten Begriffe der schwachen Generierbarkeit und schwachen Widerrufbarkeit: Protokolle, die starke Generierbarkeit oder Widerrufbarkeit voraussetzen, können meist ohne Änderung der Implementierung mit Objekten durchgeführt werden, die lediglich schwache Generierbarkeit bzw. Widerrufbarkeit bieten. Als Ergebnis ändert sich in vielen Fällen der Grad der Fairneß von F4 auf F2. Es genügt also die Implementierung eines einzigen Protokolls, um verschiedene Arten von Fairneß anzubieten. Dadurch sinkt der Aufwand für die Implementierung, was Kosten spart und die Sicherheit erhöht, da nur eine Implementierung auf ihre Korrektheit überprüft werden muß.

Aber auch in umgekehrter Weise profitiert man von der Einführung der Begriffe der schwachen Generierbarkeit und Widerrufbarkeit. Alle Protokolle, die die schwache Form dieser Objekteigenschaften voraussetzen, können auch mit stark generierbaren bzw. widerrufbaren Objekten aufgerufen werden. Da ein solches stark generierbares bzw. widerrufbares Objekt automatisch die Bedingungen der schwachen Generierbarkeit bzw. Widerrufbarkeit erfüllt, behalten alle Aussagen über die Eigenschaften des Protokolls in vollem Umfang ihre Gültigkeit.

In Abschnitt 3.2 werden optionale Erweiterungen für die Nichtabstreitbarkeit vorgeschlagen. Während die Nichtabstreitbarkeit der Herkunft nur eine Erweiterung der ausgetauschten Objekte voraussetzt, müssen für die Nichtabstreitbarkeit des Empfangs die vorhandenen Protokolle erweitert werden. Diese Protokollerweiterungen erfolgen jedoch für alle optimistischen Protokolle nach dem gleichen Schema, was für die Qualität der von

mir vorgeschlagenen Modellierung spricht und eine Implementierung der Erweiterungen für Nichtabstreitbarkeit des Empfangs deutlich erleichtert. Außerdem zeige ich, daß die Erweiterung für Nichtabstreitbarkeit des Empfangs auch eine zusätzliche Verbesserung eines Protokolls bewirken kann: Das Protokoll P5-NE für widerrufbare Objekte profitiert von dieser Erweiterung durch eine bessere Konfliktlösung.

Zum Schluß dieses Kapitels diskutiere ich, wie man die geeigneten Protokolle auswählt, wenn man die Eigenschaften der ausgetauschten Objekte kennt. Mit diesen in Abschnitt 3.3 aufgestellten Regeln kann eine Anwendung automatisch eines aus den von FlexiFair angebotenen Protokollen selektieren.

3.1 Protokolle

In diesem Abschnitt werden verschiedene aktive und optimistische Protokolle vorgestellt. Das Ziel dieser Protokolle ist es, F4-Fairneß und möglichst auch T2-Terminierung zu erreichen. Die Protokolle sind dabei bewußt generisch formuliert: Es werden beliebige digitale Objekte ausgetauscht, die je nach Protokoll noch zusätzlich die Eigenschaft der Generierbarkeit oder Widerrufbarkeit besitzen sollten. Es wird angenommen, daß eine Beschreibung der ausgetauschten Objekte und eine Methode zum Überprüfen, ob ein gegebenes Objekt einer Beschreibung entspricht, existiert. Wie solche Beschreibungen realisiert werden können wird dann im Implementierungskapitel in Abschnitt 4.3 erklärt. Für die generische Beschreibung der Protokolle sind solche Implementierungsdetails jedoch unwichtig und werden daher weggelassen, um die Protokolle möglichst einfach und verständlich zu halten. Es wird weiterhin angenommen, daß alle Nachrichten authentisch sind. Angriffe wie das Hinzufügen, Verändern oder Wiedereinspielen von Nachrichten können durch die in Abschnitt 4.1 vorgestellten Maßnahmen auf der Implementierungsebene verhindert werden.

Im folgenden wird zuerst eine von allen Protokollen verwendbare Realisierung von Modul M1 vorgestellt. In Abschnitt 3.1.2 wird ein aktives Protokoll beschrieben, bevor in den Abschnitten 3.1.3 bis 3.1.6 die optimistischen Protokolle folgen. Mit den von mir in den Abschnitten 3.1.4 bis 3.1.6 neu vorgeschlagenen Protokollen zeige ich, daß auch mit widerrufbaren Objekten optimistische Protokolle konstruiert werden können, die die gleichen Fairneßeigenschaften besitzen wie optimistische Protokolle, die nur Generierbarkeit voraussetzen.

3.1.1 Aushandeln der Protokollparameter

Bevor ein Fairneßprotokoll ausgeführt werden kann, müssen sich die beteiligten Parteien auf ein bestimmtes Protokoll und die verwendeten Parameter einigen. Dies ist gerade die Aufgabe des in Abschnitt 2.3.3 vorgestellten Moduls M1. Die in diesem Abschnitt beschriebene Implementierung I1 von Modul M1 (siehe Tabelle 3.1) kann von allen im folgenden vorgestellten Austauschprotokollen verwendet werden. Aus diesem Grund wird nur diese eine Realisierung als ein Beispiel für eine mögliche Implementierung von Modul M1 vorgeschlagen.

I1: Implementierung von Modul M1		
A $\rightarrow$ B	:	Unterstützte Protokolle und Vermittler, Beschreibung von O_B
B	:	Prüfe, ob die gelieferte Beschreibung zum vorhandenen O_B paßt (falls nein, beende das Protokoll) Wähle von beiden Parteien unterstütztes Protokoll und Vermittler aus
B $\rightarrow$ A	:	Ausgewähltes Protokoll und Vermittler V, Beschreibung von O_A
A	:	Prüfe, ob die gelieferte Beschreibung zum vorhandenen O_A paßt und ob das gewählte Protokoll und V korrekt sind (falls nein, beende das Protokoll)

Tabelle 3.1: Eine Implementierung von Modul M1 für beliebige Austauschprotokolle.

In der Implementierung I1 einigen sich die Parteien A und B auf ein Protokoll und einen Vermittler, indem A verschiedene Protokolle und Vermittler vorschlägt und B sich ein Protokoll und einen Vermittler heraussucht. Außerdem tauschen A und B die Beschreibungen der erwarteten Objekte aus. Falls sich die beiden Parteien nicht einigen können (z.B. weil die gewünschten Protokolle oder Objektbeschreibungen nicht zusammenpassen), brechen sie einfach die Protokollausführung ab. Dies ist aus Sicht beider Parteien vertretbar, da noch nichts Wertvolles ausgetauscht wurde.

3.1.2 Protokoll P1: Austausch mit aktivem Vermittler

In diesem Abschnitt beschreibe ich ein Austauschprotokoll mit aktiv beteiligtem Vermittler. Dieses Protokoll stellt keine speziellen Anforderungen an die auszutauschenden Objekte und empfiehlt sich daher als Standardprotokoll für den fairen Austausch.

Protokoll P1

Voraussetzungen: Keine, da beliebige digitale Objekte verwendet werden können.

Implementierung: I1, I2-akt, I3-akt, I4-akt, I5-akt.

Eigenschaften: Aktives Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für beide Parteien.

Im folgenden beschreibe ich zuerst die Implementierung der Module M2 bis M5, welche mit I2-akt bis I5-akt bezeichnet werden. Die Art der Benennung gibt die Nummer des implementierten Moduls (also 2 bis 5) und die Eigenschaft des implementierten Protokolls (hier: *akt*ives Protokoll) an. Anschließend diskutiere ich die Eigenschaften von Protokoll P1 und demonstriere die einfache Erweiterbarkeit eines solchen modularisierten Protokolls, indem ich eine Protokollvariante vorstelle, die den Speicheraufwand des Vermittlers reduziert.

Protokollbeschreibung

Für das Aushandeln der Protokollparameter in Modul M1 greife ich auf die in Tabelle 3.1 beschriebene Implementierung I1 zurück. Der eigentliche Austausch mit aktivem Vermittler wird dann durch die in den Tabellen 3.2 und 3.3 dargestellten Implementierungen I2-akt und I3-akt realisiert.

Implementierung I2-akt: Zur Vorbereitung des Austausches schicken beide Parteien ihre Objekte zusammen mit der Beschreibung der gewünschten Objekte zum Vermittler V, der anhand dieser Beschreibungen eine Überprüfung der gelieferten Objekte durchführt. Falls eines der Objekte nicht der Beschreibung entspricht, bricht der Vermittler den Austausch ab und informiert die Parteien A und B über diese Entscheidung.

Implementierung I3-akt: Nachdem das Vorbereiten des Austausches erfolgreich verlaufen ist, besitzt der Vermittler die gewünschten Objekte und führt den Austausch wie in Tabelle 3.3 beschrieben durch. Dabei liefert der Vermittler die gewünschten Objekte aus, d.h. A erhält O_B und B erhält O_A.

Falls beim Vorbereiten oder Durchführen des Austausches ein Fehler auftritt (z.B. weil es eine Kommunikationsunterbrechung gab oder weil nur eine Partei ihr Objekt an den Vermittler geliefert hat, die andere jedoch nicht), kann eine davon betroffene Partei $P \in \{A, B\}$ die Fehlerbehandlung mit den Modulimplementierungen I4-akt oder I5-akt starten.

Implementierung I4-akt: Falls nach einem Fehler das Weiterführen des Austausches gewünscht ist, muß die davon betroffene Partei P eine Konfliktlösung gemäß Modul M4 ausführen (siehe Tabelle 3.4). Dabei schickt P zuerst dem Vermittler sein Objekt O_P und die Beschreibung des gewünschten Objekts O_Q (Q ist der Bezeichner für die von P verschiedene Partei, also $Q \in \{A, B\} \setminus \{P\}$). Der Vermittler reagiert darauf in Abhängigkeit seines eigenen Zustands:

- Falls der Vermittler bereits mit der Auslieferung der Objekte begonnen hat, kann er das immer noch gespeicherte Objekt O_Q an P schicken und die Ausführung I4-akt erfolgreich abschließen.
- Falls der Vermittler den Austausch bereits abgebrochen hat, wird er P darüber informieren.
- Falls dem Vermittler das Objekt O_Q fehlt, kann er den Austausch nicht weiterführen und wird daher P lediglich über die Ursache benachrichtigen. Dieser kann dann mittels I5-akt eine Beendigung des Austausches erzwingen.
- Falls beide Objekte vorliegen, wird der Vermittler prüfen, ob diese den Anforderungen der Beschreibungen entsprechen. Wenn ja, dann werden die Objekte an die Parteien P und Q ausgeliefert. Wenn nein, dann bricht der Vermittler den Austausch ab und informiert beide Parteien darüber.

I2-akt: Implementierung von Modul M2 für aktiven Austausch		
A → V	:	O_A, Beschreibung von O_B
B → V	:	O_B, Beschreibung von O_A
V	:	Überprüfen von O_A und O_B
		Falls ein Objekt nicht der Beschreibung entspricht:
		V → A : Austausch abgebrochen
		V → B : Austausch abgebrochen

Tabelle 3.2: Implementierung von Modul M2 für fairer Austausch mit aktivem Vermittler.

I3-akt: Implementierung von Modul M3 für aktiven Austausch		
V → A	:	O_B
V → B	:	O_A

Tabelle 3.3: Implementierung von Modul M3 für fairer Austausch mit aktivem Vermittler.

I4-akt: Implementierung von Modul M4 für aktiven Austausch		
P → V	:	O_P, Beschreibung von O_Q
V	:	Falls der Austausch in I3-akt bereits gestartet wurde:
		V → P : O_Q
		Falls der Austausch bereits abgebrochen wurde:
		V → P : Austausch abgebrochen
		Falls O_Q noch nicht vorhanden ist:
		V → P : Fehler: Objekt O_Q fehlt
		Falls erst jetzt O_P und O_Q vorhanden sind:
		Überprüfen von O_P und O_Q
		Falls die Objekte den Beschreibungen entsprechen:
		V → P : O_Q
		V → Q : O_P
		Sonst:
		V → P : Fehler: Austausch abgebrochen
		V → Q : Fehler: Austausch abgebrochen

Tabelle 3.4: Mit dieser Implementierung von Modul M4 kann die Partei $P \in \{A, B\}$ den fairen Austausch mit aktivem Vermittler V fortsetzen.

I5-akt: Implementierung von Modul M5 für aktiven Austausch		
P → V	:	Abbrechen
V	:	falls der Austausch noch nicht durchgeführt wurde:
		V → P : Austausch abgebrochen
		sonst:
		V → P : O_Q

Tabelle 3.5: Mit dieser Implementierung von Moduls M5 kann die Partei $P \in \{A, B\}$ den fairen Austausch mit aktivem Vermittler V abbrechen.

Implementierung I5-akt: Solange die Auslieferung von O_A und O_B noch nicht begonnen hat, kann eine Partei $P \in \{A, B\}$ das Zurücksetzen des Austausches vom Vermittler fordern. Eine Implementierung dieses Moduls M5 ist in Tabelle 3.5 dargestellt. Der Vermittler wird dieser Aufforderung zum Abbruch des Austausches nur dann nachkommen, wenn der Austausch noch nicht während I3-akt oder I4-akt durchgeführt wurde. Ansonsten liefert der Vermittler das gewünschte Objekt O_Q.

Diskussion

Die Ausführungsreihenfolge der einzelnen Module für dieses Austauschprotokoll mit aktivem Vermittler ist nochmal in Abbildung 3.1 zusammengefaßt. Eine Besonderheit gegenüber der allgemeinen Ausführungsreihenfolge der Module in Abbildung 2.3 besteht darin, daß grundsätzlich keine externe Konfliktlösung nötig ist. Da der Vermittler atomar entscheidet, ob der Austausch fortgesetzt oder zurückgesetzt wird, muß entweder die Konfliktlösung durch Fortsetzen oder Zurücksetzen F4-Fairneß und T2-Terminierung garantieren. Auf eine ausführliche Analyse des Fairneßgrades verzichte ich hier, da es erstens offensichtlich ist, daß das Protokoll die Eigenschaften Wirksamkeit, F4-Fairneß und T2-Terminierung besitzt, und zweitens schon in vielen anderen Veröffentlichungen (siehe z.B. [BP90, Tyg96, ZG96, FR97]) die Sicherheit von aktiven Protokollen diskutiert wurde.

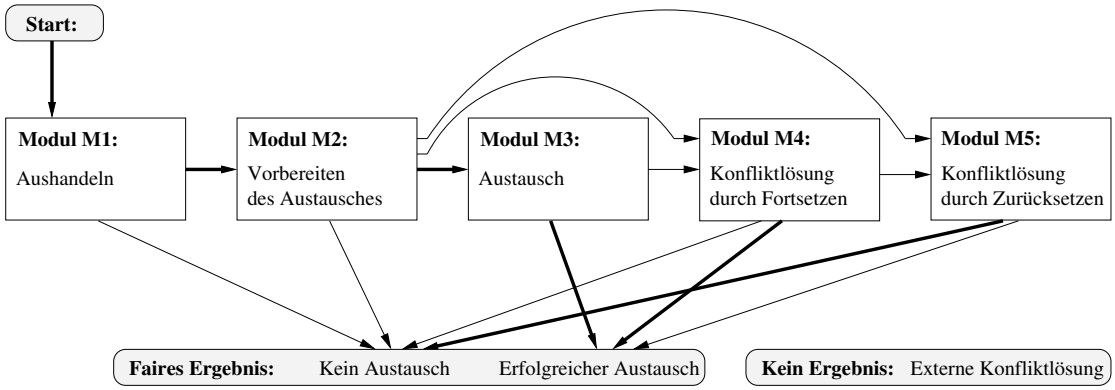

Abbildung 3.1: Mögliche Ausführungsreihenfolgen der Modulimplementierungen I1, I2-akt, I3-akt, I4-akt und I5-akt für fairen Austausch mit aktivem Vermittler.

Einer der Vorteile des Modularisierungsansatzes für Austauschprotokolle wird im folgenden Beispiel klar, wenn man das Ersetzen eines einzelnen Moduls betrachtet: Die Implementierung I3-akt hat den Nachteil, daß der Vermittler auch bei einem fehlerfreien Austausch die Objekte O_A und O_B dauerhaft speichern muß, damit eine Fehlerbehandlung durch I4-akt und I5-akt immer möglich ist. Gerade bei sehr großen Objekten kann dies hohe Kosten beim Vermittler verursachen, so daß besser die alternative Implementierung I3-akt' eingesetzt wird, die in Tabelle 3.6 dargestellt ist.

I3-akt': Alternative Implementierung von M3 für aktiven Austausch		
V $\rightarrow$ A	:	O_B
V $\rightarrow$ B	:	O_A
A $\rightarrow$ V	:	Empfangsbestätigung für O_B
B $\rightarrow$ V	:	Empfangsbestätigung für O_A
V	:	Lösche bestätigte Objekte

Tabelle 3.6: Alternative Implementierung von Modul M3 für fairer Austausch mit aktivem Vermittler. Die Empfangsbestätigungen reduzieren den Aufwand für die Speicherung der Objekte beim Vermittler.

Nachdem der Vermittler die Objekte O_A und O_B an B und A verschickt hat, wartet er noch auf Empfangsbestätigungen für diese Objekte. Nach dem Erhalt einer Empfangsbestätigung von Partei $P \in \{A, B\}$ weiß der Vermittler, daß P den Austausch erfolgreich beendet hat und somit nicht mehr die Fehlerbehandlung I4-akt oder I5-akt in Anspruch nehmen wird. Deshalb kann er das Objekt O_Q löschen. Der Vollständigkeit halber sollten schließlich noch die Implementierungen I4-akt und I5-akt angepaßt werden, so daß der Vermittler statt einem bereits bestätigten und daher nicht mehr vorhandenen Objekt die wesentlich kleinere Empfangsbestätigung ausliefert.

3.1.3 Protokoll P2: Optimistischer Austausch mit generierbaren Objekten

In diesem Abschnitt stelle ich ein generisches Protokoll für optimistisch fairen Austausch vor, welches die Generierbarkeit beider Objekte voraussetzt. Die wesentlichen Ideen für dieses Protokoll wurden in [ASW97b, ASW98a, Aso98] vorgestellt.

Protokoll P2

Voraussetzungen: O_A stark generierbar, O_B stark generierbar.

Implementierung: I1, I2-opt-gg, I3-opt-gg, I4-opt-gg, I5-opt-gg.

Eigenschaften: Optimistisches Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für beide Parteien.

Mit Hilfe des in Abschnitt 2.3.4 eingeführten Begriffs der schwachen Generierbarkeit gelingt mir die Verallgemeinerung der in [ASW98a, siehe Abb. 2 bzw. 4] angegebenen Austauschprotokolle für Vertragsunterzeichnung bzw. E-Mail-Empfangsbestätigung. Dieses P2' genannte Protokoll kann beiden Parteien schon dann F4-Fairneß und T2-Terminierung garantieren, wenn A ein schwach generierbares Objekt O_A und B ein stark generierbares O_B verwendet.

Protokoll P2'

Voraussetzungen: O_A schwach generierbar, O_B stark generierbar.

Implementierung: I1, I2-opt-gg, I3-opt-gg, I4-opt-gg für A, I4-opt-gg' für B, I5-opt-gg.

Eigenschaften: Optimistisches Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für beide Parteien.

Im folgenden beschreibe ich zuerst die Implementierungen I2-opt-gg bis I5-opt-gg (die Benennung deutet an, daß es sich um ein optimistisches Protokoll mit zwei generierbaren Objekten handelt). Anschließend diskutiere ich, wie sich schwächere Formen der Generierbarkeit auf die Fairneß und Terminierung auswirken. Dabei zeige ich, daß das Protokoll P2' mit einer geänderten Implementierung I4-opt-gg' auch F4-Fairneß und T2-Terminierung für beide Parteien garantieren kann.

Protokollbeschreibung

Für das Aushandeln des optimistisch fairen Austausches kann wieder auf die Implementierung I1 aus Tabelle 3.1 auf Seite 39 zurückgegriffen werden.

Implementierung I2-opt-gg: Die Implementierung von Modul M2 ist in Tabelle 3.7 dargestellt. Zur Vorbereitung des Austausches überprüfen beide Parteien, ob das Objekt der anderen Partei durch den Vermittler generierbar ist. Falls sich das Objekt O_A nicht als durch den Vermittler generierbar herausstellt oder keine Nachricht von A ankommt, kann B einfach den Austausch beenden. Dies ist möglich, da B noch keine für den Austausch notwendigen Informationen weitergegeben hat, so daß keine weitere Fehlerbehandlung notwendig ist.

Falls O_A generierbar ist, schickt B die notwendigen Informationen, mit denen der Vermittler O_B generieren kann. Dann prüft A, ob diese Informationen wirklich korrekt sind. Wenn A dabei einen Fehler entdeckt bzw. überhaupt keine Nachricht empfängt, muß er die Fehlerbehandlung mit der Modulimplementierung I5-opt-gg einsetzen. Ansonsten ist die Vorbereitung des Austausches beendet.

Implementierung I3-opt-gg: Der Austausch in Modul M3 (siehe Tabelle 3.8) beginnt damit, daß A sein Objekt O_A an B schickt. Falls es sich nicht um das gewünschte Objekt handelt oder keine Nachricht von A ankommt, muß B zur Fehlerbehandlung Modulimplementierung I4-opt-gg ausführen. Ein korrektes O_A beantwortet B dagegen mit O_B. Sobald A das den Anforderungen entsprechende O_B empfangen hat, ist der Austausch erfolgreich beendet. Falls A kein oder ein ungültiges O_B empfängt, muß er auf die Modulimplementierung I4-opt-gg zur Fehlerbehandlung zurückgreifen.

Implementierung I4-opt-gg: Mit der Implementierung I4-opt-gg (siehe Tabelle 3.9) kann die Partei $P \in \{A, B\}$ einen fehlerhaften oder unterbrochenen Austausch fortsetzen. Dazu sendet P sein Objekt, die Beschreibungen der auszutauschenden Objekte und falls vorhanden die Generierungsinformation für O_Q. Der Vermittler wird die Fehlerbehandlung nur dann starten, wenn noch kein Abbruch mit I5-opt-gg durchgeführt wurde. Zuerst prüft er, ob P das von Q gewünschte Objekt O_P geliefert hat. Falls sich dabei herausstellt, daß O_P nicht der geforderten Beschreibung entspricht, verweigert der Vermittler die Fortführung des Austausches. Ansonsten prüft der Vermittler zum Schluß,

I2-opt-gg: Implementierung von Modul M2 für optimistischen Austausch		
A → B	:	Generierungsinformation für O_A
B	:	Prüfe, ob O_A generierbar ist (falls nein, beende Austausch)
B → A	:	Generierungsinformation für O_B
A	:	Prüfe, ob O_B generierbar ist (falls nein, starte I5-opt-gg)

Tabelle 3.7: Modul M2 für optimistisch fairen Austausch mit zwei generierbaren Objekten.

I3-opt-gg: Implementierung von Modul M3 für optimistischen Austausch		
A → B	:	O_A
B	:	Prüfe, ob O_A der Beschreibung entspricht (falls nein, starte I4-opt-gg)
B → A	:	O_B
A	:	Prüfe, ob O_B der Beschreibung entspricht (falls nein, starte I4-opt-gg)

Tabelle 3.8: Modul M3 für optimistisch fairen Austausch mit zwei generierbaren Objekten.

I4-opt-gg: Implementierung von Modul M4 für optimistischen Austausch		
P → V	:	O_P, die Beschreibungen beider Objekte und die Generierungsinformation für O_Q
V	:	Falls der Austausch noch nicht abgebrochen wurde:
		Falls P das von Q gewünschte Objekt O_P geliefert hat:
		Falls O_Q generiert werden kann oder bereits gespeichert ist:
		V : Speichere O_P
		V → P : O_Q
		Falls O_Q nicht generiert werden kann:
		V → P : Fehler: O_Q nicht generierbar
		Falls P nicht das von Q gewünschte Objekt O_P geliefert hat:
		V → P : Fehler: Falsches O_P
		Falls der Austausch bereits abgebrochen wurde:
		V → P : Austausch abgebrochen

Tabelle 3.9: Optimistisch fairer Austausch mit zwei generierbaren Objekten: In der Implementierung von M4 möchte $P \in \{A, B\}$ mit dem Vermittler den Austausch fortsetzen.

I5-opt-gg: Implementierung von Modul M5 für optimistischen Austausch		
A → V	:	Austausch abbrechen
V	:	Falls I4-opt-gg bereits erfolgreich ausgeführt wurde:
		V → A : O_B
		Sonst:
		V → A : Austausch erfolgreich abgebrochen

Tabelle 3.10: Optimistisch fairer Austausch mit zwei generierbaren Objekten: In der Implementierung von M5 möchte A den Abbruch des Austausches durch den Vermittler erreichen.

ob er in der Lage ist, O_Q zu liefern. Dies ist genau dann möglich, wenn O_Q wirklich generierbar ist oder wenn der Vermittler O_Q bei der Ausführung von I4-opt-gg durch die Partei Q bereits gespeichert hat. Bevor der Vermittler schließlich das Objekt O_Q an P schickt, speichert er noch das Objekt O_P, so daß es bei einem späteren Aufruf der Fehlerbehandlungen durch Q bereits vorhanden ist.

Implementierung I5-opt-gg: Eine Konfliktlösung durch Zurücksetzen des Austausches ist nur für die Partei A möglich, da B entweder die weitere Ausführung während I2-opt-gg stoppen kann oder eine Fehlerbehandlung mit I4-opt-gg durchführen muß. Mit dem in Tabelle 3.10 beschriebenen Protokoll I5-opt-gg kann A den Abbruch des Austausches beim Vermittler beantragen. Wenn bereits eine Konfliktlösung mit I4-opt-gg durchgeführt wurde (z.B. weil B beim Durchführen einer Konfliktlösung schneller war), ist kein Abbruch mehr möglich. Stattdessen erhält A das gespeicherte O_B. Falls der Vermittler bisher noch keine Entscheidung für diesen Austauschvorgang getroffen hatte, wird er dem Abbruch zustimmen und A eine entsprechende Bestätigung schicken.

Diskussion

Im Unterschied zu dem in Abschnitt 3.1.2 vorgestellten Protokoll P1 mit aktivem Vermittler werden für dieses optimistische Protokoll generierbare Objekte benötigt. Um die Fairneß dieses Protokolls besser verstehen und bewerten zu können, skizziere ich im folgenden, welche Auswirkungen unterschiedlich starke Generierbarkeit der Objekte auf das Protokoll hat. Zuerst wird der Einfluß der Generierbarkeit von O_A auf die Partei B diskutiert:

1. O_A **stark generierbar:** In diesem Fall garantiert das Protokoll F4-Fairneß für die Partei B. Bei einem Fehler startet B I4-opt-gg, was eines der zwei folgenden Ergebnisse produziert:

   a) B erhält das Objekt O_A nicht, weil A den Austausch bereits mit I5-opt-gg abgebrochen hat. Dann kann aber A unmöglich O_B empfangen haben, weil B in I3-opt-gg das gewünschte Objekt immer zuerst bekommt.

   b) B erhält O_A, weil der Vermittler beim Generieren von O_A nie scheitern wird, wenn sich B korrekt verhalten hat.

   In diesen beiden Fällen sowie nach der fehlerfreien Protokollausführung terminiert B, so daß das Protokoll T2-Terminierung für B garantiert, weil keine weitere Zustandsänderung mehr möglich ist.

2. O_A **schwach generierbar:** In diesem Fall kann ebenfalls F4-Fairneß und T2-Terminierung für die Partei B garantiert werden. Der Vermittler muß jedoch ein leicht geändertes Konfliktlösungsprotokoll I4-opt-gg' für B ausführen (siehe Tabelle 3.11). Der Unterschied besteht darin, daß der Vermittler beim Generieren von O_A scheitern kann, wenn A dessen Generierbarkeit verhindert. Da aber A zu diesem Zeitpunkt garantiert nicht O_B erhalten hat, kann der Vermittler den Austausch

einfach abbrechen und so die F4-Fairneß und T2-Terminierung sicherstellen.

I4-opt-gg': Alternative Implementierung von Modul M4 für optimistischen Austausch		
B → V	:	O_B, die Beschreibungen beider Objekte und die Generierungsinformation für O_A
V	:	Falls der Austausch noch nicht abgebrochen wurde:
		Falls B das von A gewünschte Objekt O_B geliefert hat:
		Falls O_A generiert werden kann oder bereits gespeichert ist:
		V : Speichere O_B
		V → B : O_A
		Falls O_A nicht generiert werden kann:
		V → B : Austausch abgebrochen (wegen Betrug von A)
		Falls B nicht das von A gewünschte Objekt O_B geliefert hat:
		V → B : Fehler: Falsches O_B
		Falls der Austausch bereits abgebrochen wurde:
		V → B : Austausch abgebrochen

Tabelle 3.11: Alternative Implementierung von Modul M4 für optimistisch fairer Austausch mit schwach generierbarem Objekt O_A.

3. O_A **nicht generierbar:** In diesem Fall kann F4-Fairneß für B nur unter Verzicht auf T2-Terminierung erreicht werden.
   Im Unterschied zu 2. wird hier der Vermittler immer am Generieren von O_A scheitern. Dies hat zur Folge, daß der Vermittler einen Betrug von A nicht mehr eindeutig feststellen kann (z.B. kann B fälschlicherweise behaupten, daß A nichts geliefert hat). Somit ist auch ein Abbruch des Austausches nach einem Fehler beim Generieren wie in I4-opt-gg' nicht gerechtfertigt. Stattdessen muß der Vermittler für B hier wieder das normale I4-opt-gg zur Konfliktlösung verwenden. Der Vermittler kann nämlich beim Ausführen von I4-opt-gg durch B nur sicher sein, daß A noch kein O_B bekommen hat, aber er weiß nicht, ob B bereits O_A hat. Daher kann der Vermittler in I4-opt-gg nur eine Fehlermeldung liefern.
   Falls B den Austausch daraufhin beendet und A später durch den Vermittler O_B generieren läßt, wird die Fairneß verletzt. Daher muß B eine schwächere Form der Terminierung wählen, um F4-Fairneß zu erreichen. Eine Protokollvariante ohne Terminierung (also T0) besteht darin, daß B immer wieder einmal I4-opt-gg startet, um festzustellen, ob A inzwischen die Konfliktlösung mit I4-opt-gg durchgeführt und dabei O_A geliefert hat. Eine andere Variante mit T1-Terminierung ist möglich, wenn B die Ausführung des Austauschprotokolls beenden kann, aber vom Vermittler das Objekt O_A zugestellt bekommt, wenn A doch noch die Konfliktlösung mit I4-opt-gg durchführt.

Die Generierbarkeit von O_B beeinflußt dagegen die Fairneß und Terminierung für die Partei A. Dies betrifft allerdings nur die Konfliktlösung mit I4-opt-gg, da beim Abbruch mit

I5-opt-gg immer F4-Fairneß und T2-Terminierung für A garantiert wird, weil entweder A das gewünschte O_B erhält oder B nicht O_A.

1. O_B **stark generierbar:** In diesem Fall garantiert das Protokoll F4-Fairneß für die Partei A, weil der Vermittler bei der Konfliktlösung mit I4-opt-gg O_B immer erfolgreich generiert. Außerdem terminiert A in jedem Fall, so daß T2-Terminierung garantiert wird, weil nach der Terminierung keine Zustandsänderungen mehr eintreten können.

2. O_B **schwach generierbar:** Der Vermittler kann beim Generieren von O_B scheitern. Dann kann der Vermittler mit der formal gültigen Generierungsinformation für O_B aber beweisen, daß B sich nicht korrekt verhalten hat, so daß nur mit einem erfolgreichen Gerichtsverfahren F2-Fairneß für A erreicht werden kann. Eine automatische Konfliktlösung kann hier nicht mehr erfolgreich sein, da B das Objekt O_A schon erhalten haben könnte.
   Eine bessere Lösung ist möglich, falls zusätzlich O_A stark generierbar ist: A und B können ihre Rollen tauschen, so daß der unter „2. O_A schwach generierbar:“ beschriebene Fall erreicht wird, wodurch F4-Fairneß für A erreicht wird, ohne daß die Fairneß für B beeinträchtigt wird.

3. O_B **nicht generierbar:** Der Vermittler wird beim Generieren von O_B immer scheitern, solange B nicht I4-opt-gg ausführt und der Vermittler das Objekt O_B speichert. Da aber B in I3-opt-gg als erster das gewünschte Objekt erhält, wird B normalerweise keine Konfliktlösung starten. Deshalb kann in diesem Fall keine Fairneß (d.h. F0-Fairneß) für A erreicht werden, weil A auch keine Beweise für ein Gerichtsverfahren erhalten hat.
   Eine bessere Lösung ist möglich, falls zusätzlich O_A stark generierbar ist: A und B können ihre Rollen tauschen, so daß der unter „3. O_A nicht generierbar:“ beschriebene Fall eintritt, wodurch F4-Fairneß mit T0/T1-Terminierung für A erreicht wird, ohne daß die Fairneß für B beeinträchtigt wird.

Die Auswirkungen der Generierbarkeit auf die Fairneß des Austauschprotokolls P2/P2' sind in Tabelle 3.12 zusammengefaßt. Die Terminierung wird in dieser Tabelle nicht gesondert aufgeführt, weil immer T2-Terminierung für beide Parteien garantiert wird, solange nicht ausdrücklich das Gegenteil gesagt wird.

Im wesentlichen kann man beim Austauschprotokoll P2/P2' drei verschiedene Kategorien unterscheiden:

1. Wenn ein Objekt stark generierbar und das andere mindestens schwach generierbar ist, wird F4-Fairneß und T2-Terminierung für beide Parteien garantiert. Da dies gemäß den Voraussetzungen von F4 ohne irgendwelche Annahmen über eine globale Zeit bzw. Laufzeiten von Nachrichten möglich ist (in Abschnitt 2.2.2 wird schließlich ein asynchrones Systemmodell vorausgesetzt), bezeichnet man diese Protokollen auch als *asynchrone Protokolle* [ASW98a]. Andere sehr ähnliche asynchrone Protokolle findet man beispielsweise in [ZDB99, GJM99, MS01].

Generierbarkeit		Fairneß für		Anmerkung
O_A	O_B	A	B	
stark	stark	F4	F4	Beweisidee siehe auch [ASW97b]
stark	schwach	F2	F4	Tausche A und $B \Rightarrow$ F4 für A
stark	–	F0	F4	Tausche A und $B \Rightarrow$ F4 mit T0/T1 für A
schwach	stark	F4	F4[1)]	[1)] gilt für P2'
schwach	schwach	F2	F4[1)]	[1)] gilt für P2'
schwach	–	F0	F4[1)]	[1)] gilt für P2'
–	stark	F4	F4[2)]	[2)] nur T0/T1-Terminierung für B
–	schwach	F2	F4[2)]	[2)] nur T0/T1-Terminierung für B
–	–	F0	F4	

Tabelle 3.12: Einfluß der Generierbarkeit der Objekte auf die Fairneß des optimistischen Protokolls P2/P2'.

2. Wenn ein Objekt stark generierbar und das andere nicht generierbar ist, wird F4-Fairneß für beide Parteien nur dann garantiert, wenn eine Partei die abgeschwächte T0/T1-Terminierung akzeptiert.
   Um zu verhindern, daß eine Partei wegen dieser schwachen Terminierungsgarantie ewig auf das Endergebnis des Austausches wartet, wird in der Praxis meist eine Zeitbegrenzung vorgeschlagen: Der Vermittler wird nur bis zu einem festgelegten Zeitpunkt das Objekt auf Anfrage generieren. Danach lehnt er jede Anfrage zur Generierung des Objekts ab, wodurch er eine Partei benachteiligt, wenn sie sich zu spät gemeldet hat. Aus diesem Grund muß beim Festsetzen der Zeitbegrenzung sichergestellt werden, daß jede Partei den Vermittler noch problemlos vor Ablauf der Zeit anrufen kann. Dies impliziert eine möglichst lange Zeitspanne, in der der Vermittler die Fehlerbehandlung noch durchführt.
   Aus der Sicht der Partei, die eventuell erst nach Ablauf der Zeit terminieren kann, sollte die Zeitbegrenzung dagegen möglichst kurz sein. Diese widersprüchlichen Anforderungen sind das wesentliche Problem bei dieser Art von Protokollen, die auch *synchrone Protokolle* genannt werden. Weitere Beispiele für synchrone Protokolle mit einem generierbarem Objekt findet man in [ASW97a, ZG97, BDM98, Sch00].

3. Wenn beide Objekte höchstens schwach generierbar sind, muß eine Partei eine auf F2 oder sogar F0 reduzierte Fairneßgarantie akzeptieren. Solche Protokolle, die lediglich F2-Fairneß erzielen, werden zum Beispiel in [ASW97a] behandelt.

3.1.4 Protokoll P3: Optimistischer Austausch mit stark widerrufbaren und schwach generierbaren Objekten

In diesem Abschnitt stelle ich ein neues optimistisches Austauschprotokoll vor (siehe auch [Vog03]). Dieses Protokoll setzt voraus, daß das Objekt O_A stark widerrufbar und O_B schwach generierbar ist. Mit diesen Annahmen erreiche ich eine deutliche Verbesserung gegenüber vergleichbaren, bisher bekannten Protokollen: Der Grad der Fairneß

bzw. Terminierung ist höher als beim Austauschprotokoll für Geld gegen Quittung aus [ASW98a, siehe Abb. 5]. Ebenso wird eine Verbesserung der Terminierungsgarantien gegenüber dem Protokoll zum Austausch von Geld gegen Ware aus [VP99, PV99] erzielt. Außerdem stellt das neue Protokoll auch eine Verallgemeinerung gegenüber dem Protokoll in [VP99, PV99] dar, da es mit beliebigen generierbaren und widerrufbaren Objekten anstatt Geld und Ware arbeitet.

Protokoll P3

Voraussetzungen: O_A stark widerrufbar, O_B schwach generierbar.

Implementierung: I1, I2-opt-gw, I3-opt-gw, I4-opt-gw, I5-opt-gw.

Eigenschaften: Optimistisches Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für beide Parteien.

Im folgenden gebe ich zuerst die Implementierungen I2-opt-gw bis I5-opt-gw an und weise anschließend die vom Protokoll P3 erfüllten Protokolleigenschaften nach. Danach diskutiere ich die Vorteile des neuen Protokolls und untersuche die Fairneß von Protokollvariationen, die durch reduzierte Objekteigenschaften entstehen können.

Protokollbeschreibung

Für das Aushandeln des optimistisch fairen Austausches kann wieder auf die Implementierung I1 aus Tabelle 3.1 auf Seite 39 zurückgegriffen werden.

Implementierung I2-opt-gw: Das Protokoll für Modul M2 ist in Tabelle 3.13 dargestellt. Zur Vorbereitung des Austausches überprüft A, ob das Objekt O_B durch den Vermittler generierbar ist. Falls sich dieses Objekt als nicht generierbar herausstellt oder keine Nachricht von B ankommt, kann A einfach den Austausch beenden. Dies ist möglich, da A noch keine für den Austausch notwendigen Informationen weitergegeben hat, so daß keine weitere Fehlerbehandlung notwendig ist.

In allen anderen Fällen ist die Vorbereitung des Austausches beendet und es geht weiter mit Modul M3.

Implementierung I3-opt-gw: Der Austausch in Modul M3 (siehe Tabelle 3.14) beginnt damit, daß A sein Objekt O_A an B schickt. Falls es sich nicht um das gewünschte Objekt handelt oder keine Nachricht von A ankommt, muß B zur Fehlerbehandlung Modulimplementierung I5-opt-gw ausführen. Ein korrektes O_A beantwortet B dagegen mit O_B. Sobald A das den Anforderungen entsprechende O_B empfangen hat, ist der Austausch erfolgreich beendet. Falls A kein oder ein ungültiges O_B empfängt, muß er auf die Modulimplementierung I4-opt-gw zur Fehlerbehandlung zurückgreifen.

I2-opt-gw: Implementierung von Modul M2 für optimistischen Austausch		
B → A	:	Generierungsinformation für O_B
A	:	Prüfe, ob O_B generierbar ist (falls nein, beende Austausch)

Tabelle 3.13: Modul M2 für optimistisch fairen Austausch mit widerrufbarem O_A und generierbarem O_B.

I3-opt-gw: Implementierung von Modul M3 für optimistischen Austausch		
A → B	:	O_A
B	:	Prüfe, ob O_A der Beschreibung entspricht (falls nein, starte I5-opt-gw)
B → A	:	O_B
A	:	Prüfe, ob O_B der Beschreibung entspricht (falls nein, starte I4-opt-gw)

Tabelle 3.14: Modul M3 für optimistisch fairen Austausch mit widerrufbarem O_A und generierbarem O_B.

I4-opt-gw: Implementierung von Modul M4 für optimistischen Austausch		
A → V	:	O_A, die Beschreibungen beider Objekte und die Generierungsinformation für O_B
V	:	Falls A das von B gewünschte Objekt O_A geliefert hat:
		Falls B den Austausch noch nicht abgebrochen hat:
		Falls ein korrektes O_B generiert werden kann:
		V : Speichere O_A
		V → A : O_B
		Falls O_B nicht generiert werden kann:
		V : Widerrufe O_A
		V → A : Austausch abgebrochen
		Falls B den Austausch bereits abgebrochen hat:
		V : Widerrufe O_A
		V → A : Austausch abgebrochen
		Falls A nicht das von B gewünschte Objekt O_A geliefert hat:
		V → A : Fehler: Falsches O_A

Tabelle 3.15: Optimistisch fairer Austausch mit widerrufbarem O_A und generierbarem O_B: In dieser Implementierung von M4 möchte A mit dem Vermittler den Austausch fortsetzen.

I5-opt-gw: Implementierung von Modul M5 für optimistischen Austausch		
B → V	:	Austausch abbrechen
V	:	Falls A bereits erfolgreich die Konfliktlösung mit I4-opt-gw durchgeführt hat:
		V → B : O_A
		Sonst:
		V → B : Austausch erfolgreich abgebrochen

Tabelle 3.16: Optimistisch fairer Austausch mit widerrufbarem O_A und generierbarem O_B: In der Implementierung von M5 möchte B den Abbruch des Austausches durch den Vermittler erreichen.

Implementierung I4-opt-gw: Mit Modul M4 (siehe Tabelle 3.15) kann die Partei A einen fehlerhaften oder unterbrochenen Austausch fortsetzen. Dazu sendet A sein widerrufbares Objekt O_A, die Beschreibungen der auszutauschenden Objekte und die Generierungsinformation für O_B. Der Vermittler wird die Fehlerbehandlung nur dann starten, wenn A das richtige Objekt O_A geliefert hat. Falls B bereits den Abbruch mit I5-opt-gw durchgeführt hat, wird der Vermittler sicherheitshalber auch O_A widerrufen und A über den Abbruch des Austausches informieren. Ansonsten prüft der Vermittler, ob er O_B generieren kann. Falls ja, so speichert er noch das Objekt O_A, damit es bei einem späteren Aufruf der Fehlerbehandlungen durch B vorhanden ist, und schickt das Objekt O_B an A. Falls das Generieren fehlschlägt, wird der Vermittler O_A widerrufen und damit den Austausch abbrechen.

Implementierung I5-opt-gw: Eine Konfliktlösung durch Zurücksetzen des Austausches ist nur für die Partei B möglich, da A entweder die weitere Ausführung während I2-opt-gw stoppen kann oder eine Fehlerbehandlung mit I4-opt-gw durchführen muß. Mit dem in Tabelle 3.16 beschriebenen Protokoll I5-opt-gw kann B den Abbruch des Austausches beim Vermittler beantragen. Wenn A bereits eine erfolgreiche Konfliktlösung mit I4-opt-gw durchgeführt hat, ist kein Abbruch mehr möglich und der Vermittler liefert das gespeicherte O_A. Wenn B als erster die Konfliktlösung startet, wird der Vermittler dem Abbruch zustimmen, so daß A das Objekt O_B nicht mehr durch den Vermittler generieren lassen kann.

Nachweis der Protokolleigenschaften

Da es sich bei dem hier vorgestellten Protokoll P3 um ein neues Protokoll handelt, müssen zuerst dessen Eigenschaften untersucht und bewertet werden. Im Unterschied zum ersten optimistischen Protokoll in Abschnitt 3.1.3 setzt dieses Protokoll ein stark widerrufbares und ein schwach generierbares Objekt voraus. In der folgenden Analyse der Protokolleigenschaften weise ich nach, daß auch ohne ein stark generierbares Objekt die gleichen Fairneß- und Terminierungseigenschaften erzielt werden können, wie in Protokoll P2 bzw. P2'. Die nachgewiesenen Eigenschaften gelten aufgrund der Definitionen der Generierbarkeit automatisch auch für den Fall, daß O_B stark generierbar ist.

Wirksamkeit: Wenn die Modulimplementierungen I1, I2-opt-gw und I3-opt-gw ohne Betrug und ohne Fehler bis zum Ende durchgeführt werden, dann erhält jede Partei das gewünschte Objekt.

Terminierung: Die Terminierung ist für A und B immer möglich, weil

- beide Parteien während I1 den Austausch jederzeit beenden können,
- A während I2-opt-gw den Austausch jederzeit beenden kann,
- B während I2-opt-gw auftretende Fehler ignorieren kann und einfach mit I3-opt-gw weitermacht,
- A während I3-opt-gw jederzeit I4-opt-gw zur Konfliktlösung starten kann und

- B während I3-opt-gw
  - entweder I5-opt-gw zur Konfliktlösung starten kann
  - oder nach dem Empfang von O_A den Austausch beenden kann (z.B. falls ein Fehler beim Senden von O_B auftritt).

Die Konfliktlösungsprotokolle I4-opt-gw und I5-opt-gw ermöglichen ehrlichen Parteien A und B immer die Terminierung, da der Vermittler den Austausch entweder erfolgreich zu Ende führt oder durch einen Abbruch beendet. Dies geschieht auch immer nach endlicher Zeit, da der Vermittler gemäß den Systemannahmen stets eine Antwort liefert, die irgendwann ankommt.

Das Protokoll erreicht somit entweder T1- oder T2-Terminierung. Angenommen, es würde sich nur um T1-Terminierung handeln: Dann kann nur durch das Generieren oder Widerrufen von Objekten der Zustand einer bereits terminierten Partei geändert werden. Bei Generieren von O_B in I4-opt-gw' wartet A immer mit dem Terminieren, bis er O_B erhält. Beim Widerrufen von O_A gibt es zwei mögliche Ursachen: Im ersten Fall hat sich B nicht korrekt verhalten und die Generierbarkeit von O_B verhindert. Für die Terminierung ist dieser Fall irrelevant, weil diese Eigenschaft nur für ehrliche Parteien zugesichert werden muß. Im zweiten Fall wird das Widerrufen von O_A durch die vorherige Ausführung von I5-opt-gw verursacht. Eine sich korrekt verhaltende Partei B ist dann immer im Zustand „B hat O_A nicht“, so daß sich durch das Widerrufen von O_B der Zustand von B nicht ändern wird.

Damit ist nachgewiesen, daß sich der Zustand einer terminierten Partei nicht mehr ändern kann, was T2-Terminierung für beide Parteien bedeutet.

Fairneß: Es muß gezeigt werden, daß das Protokoll Fairneß sowohl für A als auch für B garantiert.
Die Fairneß für A erfordert, daß sich A nach seiner Terminierung immer in einem fairen Zustand befindet. Wenn A bei der Terminierung das Objekt O_B hat, so wird sich das aufgrund der T2-Terminierung nicht mehr ändern, was Fairneß für A garantiert. Wenn A dagegen ohne das Objekt O_B terminiert, muß es sich um einen der folgenden beiden Fälle handeln:

1. Wenn A in I2-opt-gw ohne O_B terminiert, dann kann B weder in I3-opt-gw noch in I5-opt-gw O_A erhalten.
2. Wenn A in I4-opt-gw ohne O_B terminiert, dann hat der Vermittler vorher bereits O_A widerrufen, so daß B dieses Objekt nicht mehr erhalten kann.

Daher garantiert das Protokoll auf jeden Fall Fairneß für A. Für den Beweis der umgekehrten Aussage (d.h. die Fairneß für B) müssen die Terminierungszustände von B untersucht werden. Wenn eine ehrliche Partei B bei der Terminierung das Objekt O_A hat, dann ändert sich das aufgrund der T2-Terminierung nicht mehr, was Fairneß für B garantiert. Wenn B dagegen ohne das Objekt O_A terminiert, so muß B nach einem Fehler in I3-opt-gw die Konfliktlösung I5-opt-gw ausgeführt

haben. Dann kann A weder in I3-opt-gw noch in I4-opt-gw das Objekt O_B erhalten, so daß das Protokoll für B fair ist.

Damit ist bewiesen, daß das Protokoll Fairneß für die Parteien A und B garantiert. Genauer gesagt handelt es sich um F4-Fairneß, da die Konfliktlösung automatisch ohne Gerichtsverfahren durchgeführt wird und die Beteiligung der jeweils anderen Partei an der Konfliktlösung nicht notwendig ist.

Nichtabstreitbarkeit: In dieser einfachen Form von P3 ist keine Art von Nichtabstreitbarkeit vorgesehen. Es können jedoch die Erweiterungen aus Abschnitt 3.2 angewendet werden: Das Protokoll kann Nichtabstreitbarkeit der Herkunft garantieren, wenn die Objekte O_A bzw. O_B so definiert sind, daß sie aus dem eigentlichen Objekt und einer Signatur der sendenden Partei bestehen. Nichtabstreitbarkeit des Empfangs erfordert eine Änderung des Protokolls, welche in Abschnitt 3.2.2 vorgestellt wird.

Diskussion

Das Protokoll P3 erfüllt die gleichen Fairneßeigenschaften wie das in Abschnitt 3.1.3 beschriebene Protokoll P2/P2'. Demgegenüber stellt das hier neu eingeführte Protokoll P3 eine wesentliche Neuerung dar, weil optimistische Protokolle mit vergleichbaren Fairneßgarantien (z.B. [GJM99, ZDB99, MS01]) bisher *immer* auf den Ideen von [ASW98a] aufbauen und daher mindestens ein stark generierbares Objekt voraussetzen. Im Gegensatz dazu genügt für das neue Protokoll P3 schon ein schwach generierbares Objekt zusammen mit einem stark widerrufbaren Objekt.

Ein weiterer Vorteil des neuen Protokolls liegt in seiner besonders einfachen Struktur:

- Wenn kein Fehler auftritt, müssen in den Modulen M2 und M3 lediglich 3 Nachrichten zwischen A und B geschickt werden, um einen fairen Austausch zu realisieren.
- Die Konfliktlösung ist besonders einfach, da es für jede Partei genau eine Implementierung davon gibt: Partei A verwendet I4-opt-gw und Partei B I5-opt-gw.

Dagegen benötigt das Protokoll P2/P2' immer 4 Nachrichten in den Modulen M2 und M3 und besitzt mit I4-opt-gg und I5-opt-gg zwei Konfliktlösungsmöglichkeiten für die Partei A, was die Konfliktlösung deutlich komplizierter macht. So enthielt die erste Veröffentlichung [ASW98a] des Protokolls aus 3.1.3 noch zahlreiche Fehler (siehe z.B. [ZDB00, BK00]), die meiner Meinung nach im wesentlichen auf die komplizierte Konfliktlösung zurückzuführen sind. Dagegen ist die einfache Konfliktlösung des neuen Protokolls ein großer Vorteil, weil sich die Korrektheit leichter zeigen läßt und es weniger Fehlermöglichkeiten bei der Implementierung gibt.

Protokollvariationen. Als Abschluß der Diskussion untersuche ich, wie sich Änderungen der Objekteigenschaften auf die Fairneßgarantien von P3 auswirken:

Wenn O_A stark widerrufbar ist, dann kann A mit der Konfliktlösung in I4-opt-gw, wie soeben gezeigt, F4-Fairneß für sich herstellen. Bei schwacher Widerrufbarkeit von O_A

kann das Widerrufen in I4-opt-gw scheitern, was jedoch beweist, daß B das Objekt O_A erhalten hat. Daher wird immerhin F2-Fairneß für A erreicht, wenn A mit diesem Beweis ein Gerichtsverfahren startet, in dem die Fairneß für A doch noch garantiert werden kann. Falls O_A nicht widerrufbar ist, kann B frühzeitig I5-opt-gw aufrufen und so jede spätere Konfliktlösung in I4-opt-gw unterbinden. Daher gibt in diesem Fall keine Beweise für die Benachteiligung von A.

Wenn O_B stark oder schwach generierbar ist, dann wird nach dem Generieren von O_B immer O_A gespeichert, was der Partei B F4-Fairneß durch anschließendes Aufrufen von I5-opt-gw garantiert. Wenn B zuerst I5-opt-gw aufgerufen hat, wird O_B nie generiert, so daß die F4-Fairneß für B gewährleistet ist.

Falls O_B überhaupt nicht generierbar ist, kann I4-opt-gw keine Fairneß für die Partei B gewährleisten. Da bei einem noch nicht abgebrochenen Austausch das Generieren von O_B in I4-opt-gw fehlschlägt, wird O_A widerrufen. Daher wird B unfair behandelt, wenn O_A stark oder schwach widerrufbar ist.

Die Auswirkungen der verschiedenen Arten von Generierbarkeit und Widerrufbarkeit auf die Fairneß von P3 sind in Tabelle 3.17 zusammengefaßt. Die Terminierung wird hier nicht berücksichtigt, da in jedem Fall T2-Terminierung für beide Parteien erreicht wird.

Widerruf-barkeit O_A	**Generier-barkeit O_B**	**Fairneß für** A	B	**Anmerkung**
stark	stark	F4	F4	
stark	schwach	F4	F4	
stark	–	F4	F0	
schwach	stark	F2	F4	Verwende P4 $\Rightarrow$ F4 für A
schwach	schwach	F2	F4	
schwach	–	F2	F0	
–	stark	F0	F4	Verwende P2 $\Rightarrow$ F4 für A, T0/1 für B
–	schwach	F0	F4	Verwende P2 $\Rightarrow$ F2 für A, T0/1 für B
–	–	F0	F4	

Tabelle 3.17: Einfluß der Widerrufbarkeit und Generierbarkeit auf die Fairneß von Protokoll P3.

3.1.5 Protokoll P4: Optimistischer Austausch mit stark generierbaren und schwach widerrufbaren Objekten

In diesem Abschnitt stelle ich ein weiteres neues optimistisches Austauschprotokoll vor. Dieses Protokoll setzt voraus, daß das Objekt O_A schwach widerrufbar und O_B stark generierbar ist. Im Prinzip handelt es sich bei Protokoll P4 um eine Variation des im vorherigen Abschnitt vorgestellten Protokoll P3. Dieses Protokoll P3 hatte jedoch ein stark widerrufbares O_A und ein schwach generierbares O_B vorausgesetzt.

Protokoll P4

Voraussetzungen: O_A schwach widerrufbar, O_B stark generierbar.

Implementierung: I1, I2-opt-gw, I3-opt-gw, I4-opt-gw', I5-opt-gw.

Eigenschaften: Optimistisches Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für beide Parteien.

Im folgenden beschreibe ich zuerst die neue Implementierung I4-opt-gw' von Modul M4 und weise dann die Eigenschaften des Protokolls P4 nach. Abschließend diskutiere ich die Vorteile dieses Protokolls und untersuche, wie sich schwächere Objekteigenschaften auf die Fairneß auswirken.

Protokollbeschreibung

Die Implementierung der Module M1, M2, M3 und M5 ist die gleiche wie in Protokoll P3 und wurde in Abschnitt 3.1.4 vorgestellt. Lediglich für Modul M4 wird eine geänderte Implementierung benötigt.

Implementierung I4-opt-gw': Mit Modul M4 (siehe Tabelle 3.18) kann die Partei A einen fehlerhaften oder unterbrochenen Austausch fortsetzen. Dazu sendet A sein widerrufbares Objekt O_A, die Beschreibungen der auszutauschenden Objekte und die Generierungsinformation für O_B. Der Vermittler wird die Fehlerbehandlung nur dann starten, wenn A das richtige Objekt O_A geliefert hat.

Falls B den Abbruch noch nicht mit I5-opt-gw durchgeführt hat, wird der Vermittler O_A für spätere Anfragen von B abspeichern, O_B generieren und an A ausliefern. Falls B bereits den Abbruch mit I5-opt-gw durchgeführt hat, prüft der Vermittler, ob er O_A widerrufen kann. Ist der Vermittler erfolgreich, informiert er die Partei A über das Widerrufen von O_A. Andernfalls folgt aus der schwachen Widerrufbarkeit von O_A, daß B dieses Objekt doch erhalten hat, obwohl er einen Abbruch veranlaßt hat. Daher löst der Vermittler diesen Konfliktfall durch das Generieren und Ausliefern von O_B.

Nachweis der Protokolleigenschaften

Da es sich bei dem hier vorgestellten Protokoll P4 um ein neues Protokoll handelt, müssen zuerst dessen Eigenschaften untersucht werden. Danach werden die Auswirkungen von Protokollvariationen diskutiert.

Wirksamkeit: Wenn die Modulimplementierungen I1, I2-opt-gw und I3-opt-gw ohne Betrug und ohne Fehler bis zum Ende durchgeführt werden, dann erhält jede Partei das gewünschte Objekt.

Terminierung: Die Terminierung ist für A und B immer möglich, weil

- beide Parteien während I1 den Austausch jederzeit beenden können,
- A während I2-opt-gw den Austausch jederzeit beenden kann,
- B während I2-opt-gw auftretende Fehler ignorieren kann und einfach mit I3-opt-gw weitermacht,

I4-opt-gw': Implementierung von Modul M4 für optimistischen Austausch	
A → V :	O_A, die Beschreibungen beider Objekte und die Generierungsinformation für O_B
V :	Falls A das von B gewünschte Objekt O_A geliefert hat:
	Falls B den Austausch noch nicht abgebrochen hat:
	V : Generiere O_B, speichere O_A
	V → A : O_B
	Falls B den Austausch bereits abgebrochen hat:
	Falls O_A widerrufen werden kann:
	V : Widerrufe O_A
	V → A : Austausch erfolgreich abgebrochen
	Falls O_A nicht widerrufen werden kann:
	V : Generiere O_B
	V → A : O_B
	Falls A nicht das von B gewünschte Objekt O_A geliefert hat:
	V → A : Fehler: Falsches O_A

Tabelle 3.18: Optimistisch fairer Austausch mit schwach widerrufbarem O_A und stark generierbarem O_B: In dieser Implementierung von M4 möchte A den Austausch fortsetzen.

- A während I3-opt-gw jederzeit I4-opt-gw' zur Konfliktlösung starten kann und
- B während I3-opt-gw
  - entweder I5-opt-gw zur Konfliktlösung starten kann
  - oder nach dem Empfang von O_A den Austausch beenden kann (z.B. falls ein Fehler beim Senden von O_B auftritt).

Die Konfliktlösungsprotokolle I4-opt-gw' und I5-opt-gw ermöglichen ehrlichen Parteien A und B immer die Terminierung, da der Vermittler den Austausch entweder erfolgreich zu Ende führt oder durch einen Abbruch beendet. Dies geschieht auch immer nach endlicher Zeit, da der Vermittler gemäß den Systemannahmen stets eine Antwort liefert, die irgendwann ankommt.

Das Protokoll erreicht somit entweder T1- oder T2-Terminierung. Angenommen, es würde sich nur um T1-Terminierung handeln: Dann kann nur durch das Generieren oder Widerrufen von Objekten der Zustand einer bereits terminierten Partei geändert werden. Bei Generieren von O_B in I4-opt-gw' wartet A immer mit dem Terminieren, bis er O_B erhält. Das Widerrufen von O_A in I4-opt-gw' erfolgt nur nach dem Abbruch des Austausches mit I5-opt-gw. Dann befindet sich eine sich korrekt verhaltende Partei B jedoch bereits im Zustand „B hat O_A nicht“, so daß sich der Zustand von B durch das Widerrufen von O_A nicht mehr ändern wird.

Damit ist nachgewiesen, daß sich der Zustand einer terminierten Partei nicht mehr ändern kann, was T2-Terminierung für beide Parteien bedeutet.

Fairneß: Es muß gezeigt werden, daß das Protokoll Fairneß sowohl für A als auch für B garantiert.
Die Fairneß für A erfordert, daß sich A nach seiner Terminierung immer in einem fairen Zustand befindet. Wenn A bei der Terminierung das Objekt O_B hat, so wird sich das aufgrund der T2-Terminierung nicht mehr ändern, was Fairneß für A garantiert. Wenn A dagegen ohne das Objekt O_B terminiert, muß es sich um einen der folgenden beiden Fälle handeln:

1. Wenn A in I2-opt-gw ohne O_B terminiert, dann kann B weder in I3-opt-gw noch in I5-opt-gw O_A erhalten.
2. Wenn A in I4-opt-gw' ohne O_B terminiert, dann hat der Vermittler vorher bereits O_A widerrufen, so daß B dieses Objekt nicht mehr erhalten kann.

Daher garantiert das Protokoll auf jeden Fall Fairneß für A. Für den Beweis der umgekehrten Aussage müssen die Terminierungszustände von B untersucht werden. Wenn eine ehrliche Partei B bei der Terminierung das Objekt O_A hat, dann ändert sich das aufgrund der T2-Terminierung nicht mehr, was Fairneß für B garantiert. Wenn B dagegen ohne das Objekt O_A terminiert, so muß B nach einem Fehler in I3-opt-gw die Konfliktlösung I5-opt-gw ausgeführt haben. Dann kann A aber weder in I3-opt-gw noch in I4-opt-gw' das Objekt O_B erhalten:

1. In I3-opt-gw wird B das Objekt O_B nicht senden, da er vorher einen Fehler bemerkt hat und I5-opt-gw gestartet hat.
2. In I4-opt-gw' wird entweder kein O_B geliefert oder das Scheitern des Widerrufens von O_A beweist, daß B doch O_A hat.

Also ist das Protokoll auf jeden Fall fair für B. Damit ist bewiesen, daß das Protokoll Fairneß für die Parteien A und B garantiert. Es handelt sich hier wieder um F4-Fairneß, da die Konfliktlösung automatisch ohne Gerichtsverfahren durchgeführt wird und die Beteiligung der jeweils anderen Partei an der Konfliktlösung nicht notwendig ist.

Nichtabstreitbarkeit: In dieser einfachen Form von P4 ist keine Art von Nichtabstreitbarkeit vorgesehen. Es können jedoch die Erweiterungen aus Abschnitt 3.2 angewendet werden: Das Protokoll kann Nichtabstreitbarkeit der Herkunft garantieren, wenn die Objekte O_A bzw. O_B so definiert sind, daß sie aus dem eigentlichen Objekt und einer Signatur der sendenden Partei bestehen. Nichtabstreitbarkeit des Empfangs erfordert eine Änderung des Protokolls, welche in Abschnitt 3.2.2 vorgestellt wird.

Diskussion

Das Protokoll P4 erfüllt also die gleichen Fairneßeigenschaften wie die optimistischen Protokolle P2, P2' und P3. Ebenso wie P3 stellt das hier neu eingeführte Protokoll

P4 einen Fortschritt dar, weil es nicht auf den bereits bekannten Ideen von [ASW98a] aufbaut.

Eine weitere Gemeinsamkeit mit P3 ist die einfachen Struktur von P4:

- Wenn kein Fehler auftritt, müssen in den Modulen M2 und M3 lediglich 3 Nachrichten zwischen A und B geschickt werden, um einen fairen Austausch zu realisieren.
- Die Konfliktlösung ist besonders einfach, da es für jede Partei genau eine Implementierung davon gibt: Partei A verwendet I4-opt-gw' und Partei B I5-opt-gw.

Wie bereits in Abschnitt 3.1.4 diskutiert, ist dies eine deutliche Verbesserung bzw. Vereinfachung gegenüber Protokoll P2/P2'.

Protokollvariationen. Als Abschluß der Diskussion untersuche ich, wie sich Änderungen der Objekteigenschaften auf die Fairneßgarantien von P4 auswirken:

Starke oder schwache Widerrufbarkeit von O_A stellen sicher, daß ein von B in I5-opt-gw zurecht gewählter Abbruch des Austausches in I4-opt-gw' nicht mehr revidiert wird. Dadurch ist in diesem Fall immer F4-Fairneß für B garantiert. Falls O_A nicht widerrufbar ist, wird B in I4-opt-gw' möglicherweise benachteiligt. Nach einem Abbruch des Austausches in I5-opt-gw wird das Widerrufen von O_A in I4-opt-gw' scheitern, was laut Protokollbeschreibung zum Generieren von O_B führt. Daher wird B bei starker oder schwacher Generierbarkeit von O_B ohne Widerrufbarkeit von O_A unfair behandelt.

Wenn O_B stark generierbar ist, dann wird A immer mit der Konfliktlösung I4-opt-gw' Erfolg haben und so F4-Fairneß sicherstellen. Bei schwacher Generierbarkeit von O_B kann die Konfliktlösung in I4-opt-gw' scheitern, wenn B das Generieren von O_B verhindert. Da der Vermittler diesen Betrug immer erkennen wird, besteht noch die Möglichkeit einer Konfliktlösung außerhalb des Systems, was F2 für A entspricht. Falls O_B überhaupt nicht generierbar ist, kann I4-opt-gw' keine Fairneß für die Partei A gewährleisten, da auch keine Beweise für ein Gerichtsverfahren vorhanden sind.

Die Auswirkungen der verschiedenen Arten von Generierbarkeit und Widerrufbarkeit auf die Fairneß von P4 sind in Tabelle 3.19 zusammengefaßt. Die Terminierung wird hier nicht berücksichtigt, da in jedem Fall T2-Terminierung für beide Parteien erreicht wird.

3.1.6 Protokoll P5: Optimistischer Austausch mit widerrufbaren Objekten

In diesem Abschnitt schlage ich ein neues Protokoll vor, das nur mit widerrufbaren Objekten arbeitet und keine Generierbarkeit voraussetzt. Dabei zeigt sich, daß das Widerrufen im Vergleich zum Generieren von Objekten eine etwas schwächere Lösung ist. Das Protokoll garantiert zwar F4-Fairneß für beide Parteien, aber für die Partei B wird lediglich T1-Terminierung in der Ausprägung T1– zugesichert, was in vielen Szenarien nicht ausreichend ist. Dieses Problem kann jedoch durch die Erweiterung in Abschnitt 3.2.2 deutlich abgemildert werden.

Protokoll P5

Widerruf-barkeit O_A	Generier-barkeit O_B	Fairneß für A	Fairneß für B	Anmerkung
stark	stark	F4	F4	
stark	schwach	F2	F4	Verwende P3 ⇒ F4 für A
stark	–	F0	F4	
schwach	stark	F4	F4	
schwach	schwach	F2	F4	
schwach	–	F0	F4	
–	stark	F4	F0	Verwende P2 ⇒ F4 mit T0/T1 für B
–	schwach	F2	F0	Verwende P2 ⇒ F4 mit T0/T1 für B
–	–	F0	F4	

Tabelle 3.19: Einfluß der Widerrufbarkeit und Generierbarkeit auf die Fairneß von Protokoll P4.

Voraussetzungen: O_A stark widerrufbar und O_B schwach widerrufbar.

Implementierung: I1, I3-opt-ww, I5-opt-ww.

Eigenschaften: Optimistisches Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für A, T1-Terminierung für B.

Da das Protokoll P5 starke und schwache Widerrufbarkeit verlangt, ist es natürlich gleichermaßen für zwei stark widerrufbare Objekte geeignet.

Im folgenden wird zuerst das Protokoll vorgestellt und dann werden dessen Eigenschaften nachgewiesen. Danach werden die Anwendungsgebiete dieses Protokolls diskutiert, sowie Protokollvarianten mit reduzierten Objekteigenschaften untersucht.

Protokollbeschreibung

Für das Aushandeln in Modul M1 kann wieder auf die Implementierung I1 aus Tabelle 3.1 auf Seite 39 zurückgegriffen werden. Nach dem Aushandeln ist bei diesem Protokoll keine Vorbereitung des Austausches in Modul M2 nötig. Stattdessen kann der Austausch in M3 sofort gestartet werden. Eine weitere Besonderheit des Protokolls besteht darin, daß für die Konfliktlösung keine Implementierung des Moduls M4 existiert.

Implementierung I3-opt-ww: In dieser Implementierung von Modul M3 (siehe Tabelle 3.20) erwartet B zuerst das Objekt O_A. Falls es nicht ankommt oder fehlerhaft ist, kann B einfach den Austausch beenden. Ein korrektes O_A beantwortet B dagegen mit seinem Objekt O_B. Wenn A nichts oder ein falsches O_B erhält, wird er die Konfliktlösung mit I5-opt-ww durchführen.

Implementierung I5-opt-ww: In I5-opt-ww (siehe Tabelle 3.21) wird der Vermittler falls möglich beide Objekte widerrufen, um so eine eventuelle Benachteiligung von A zu korrigieren. Wegen der schwachen Widerrufbarkeit von O_B kann dies jedoch fehlschlagen,

so daß der Vermittler nur eine Fehlermeldung zurückgibt, weil A nachweislich das Objekt O_B erhalten hat.

I3-opt-ww: Implementierung von Modul M3 für optimistischen Austausch		
A → B	:	O_A
B	:	Prüfe, ob O_A der Beschreibung entspricht (falls nein, beende Austausch)
B → A	:	O_B
A	:	Prüfe, ob O_B der Beschreibung entspricht (falls nein, starte I5-opt-ww)

Tabelle 3.20: Modul M3 für optimistisch fairen Austausch mit widerrufbarem O_A und widerrufbarem O_B.

I5-opt-ww: Implementierung von Modul M5 für optimistischen Austausch		
A → V	:	Austausch abbrechen
V	:	Falls O_B widerrufen werden kann:
	V :	Widerrufe O_B und O_A
	V → A :	Austausch erfolgreich abgebrochen
	Sonst:	
	V → A :	Fehler: Austausch war erfolgreich

Tabelle 3.21: Optimistisch fairer Austausch mit widerrufbarem O_A und widerrufbarem O_B: In der Implementierung von M5 möchte A den Abbruch des Austausches durch den Vermittler erreichen.

Nachweis der Protokolleigenschaften

Wirksamkeit: Wenn die Modulimplementierungen I1 und I3-opt-ww ohne Betrug und ohne Fehler bis zum Ende durchgeführt werden, dann erhält jede Partei das gewünschte Objekt.

Terminierung: Die Terminierung ist für A und B immer möglich, weil

- beide Parteien während I1 den Austausch jederzeit beenden können,
- B während I3-opt-ww den Austausch jederzeit beenden kann und
- A während I3-opt-ww jederzeit I5-opt-ww zur Konfliktlösung starten kann.

Das Konfliktlösungsprotokoll I5-opt-ww ermöglicht einer ehrlichen Partei A immer die Terminierung, da der Vermittler den Austausch durch einen Abbruch beendet. Dies geschieht auch immer nach endlicher Zeit, da der Vermittler gemäß den Systemannahmen stets eine Antwort liefert, die irgendwann ankommt.

Das Protokoll erreicht somit entweder T1- oder T2-Terminierung. T1-Terminierung kann in diesem Protokoll nur durch das Widerrufen von Objekten einer bereits terminierten Partei verursacht werden. Da nur die Partei A einen Widerruf veranlassen darf, kann es nach der Terminierung von A keine Zustandsänderung mehr geben,

was T2-Terminierung für A bedeutet. Nach der Terminierung von B in I3-opt-ww kann jedoch die Partei A einen Widerruf von O_A auslösen. Aus diesem Grund erzielt das Protokoll nur T1-Terminierung für B. Genauer gesagt handelt es sich hier lediglich um die Terminierung der Form T1–.

Fairneß: Es muß gezeigt werden, daß das Protokoll Fairneß sowohl für A als auch für B garantiert:
Die Fairneß für A erfordert, daß sich A nach seiner Terminierung immer in einem fairen Zustand befindet. Wenn A bei der Terminierung das Objekt O_B hat, so wird sich das aufgrund der T2-Terminierung nicht mehr ändern, was Fairneß für A garantiert. Wenn A dagegen ohne das Objekt O_B terminiert, dann hat er vorher erfolgreich I5-opt-ww ausgeführt, wodurch O_A widerrufen wurde. Daher garantiert das Protokoll auf jeden Fall Fairneß für A.
Für den Beweis der umgekehrten Aussage müssen die Terminierungszustände von B untersucht werden. Wenn B bei der Terminierung das Objekt O_A hat, kann das entweder so bleiben oder A widerruft mit I5-opt-ww doch noch das Objekt O_A. Da der Vermittler zuerst den Widerruf von O_B durchführt und dann erst den von O_A, wird sich die bereits terminierte Partei B immer in einem fairen Zustand befinden. Wenn B dagegen ohne das Objekt O_A terminiert, so hat B entweder in I3-opt-gw O_B nicht gesendet oder A hat vor der Terminierung von B die Konfliktlösung I5-opt-gw ausgeführt und beide Objekte widerrufen. Daher ist das Protokoll auf jeden Fall fair für B.
Damit ist bewiesen, daß das Protokoll Fairneß für die Parteien A und B garantiert. Es handelt sich hier wieder um F4-Fairneß, da die Konfliktlösung automatisch ohne Gerichtsverfahren durchgeführt wird und die Beteiligung der jeweils anderen Partei an der Konfliktlösung nicht notwendig ist.

Nichtabstreitbarkeit: In dieser einfachen Form von P5 ist keine Art von Nichtabstreitbarkeit vorgesehen. Es können jedoch die Erweiterungen aus Abschnitt 3.2 angewendet werden: Das Protokoll kann Nichtabstreitbarkeit der Herkunft garantieren, wenn die Objekte O_A bzw. O_B so definiert sind, daß sie aus dem eigentlichen Objekt und einer Signatur der sendenden Partei bestehen. Nichtabstreitbarkeit des Empfangs erfordert eine Änderung des Protokolls, welche in Abschnitt 3.2.2 ausführlich diskutiert wird.

Diskussion

Aufgrund der sehr schwachen Terminierungsgarantie für B scheint das Protokoll P5 für viele Anwendungen ungeeignet zu sein. Trotzdem existieren einige wenige Anwendungsbeispiele, in denen dieses Protokoll sinnvoll einsetzbar ist.

Ein Beispiel wäre der Verkauf eines Kinotickets. Dabei müßte dann das Geld stark widerrufbar sein. Das Kinoticket dagegen kann nur schwach widerrufbar sein, weil es, nachdem die Leistung erbracht wurde, nicht mehr widerrufen werden kann. Sei also A der Käufer und B der Verkäufer des Kinotickets. Dann hat der Verkäufer B bei der Terminierung den Nachteil, daß A den Kauf irgendwann widerrufen kann. Nachdem der

entsprechende Film vorgeführt wurde, hat B jedoch die Gewißheit, ob A den Film gesehen hat oder nicht, und ob A den Kauf noch widerrufen kann oder nicht. Nach dem Anschauen des Films kann A sein Geld nämlich nicht mehr zurückerhalten. Daher kann B terminieren, ohne weitere Änderungen bezüglich des Austausches befürchten zu müssen.

Ein anderes Beispiel, wie man trotz des Terminierungsproblems das Protokolls P5 sinnvoll einsetzen kann, wird in Abschnitt 3.2.2 beschrieben. Durch die Erweiterung um Nichtabstreitbarkeit des Empfangs erhält B starke Hinweise darauf, ob sich der Zustand des Austausches noch ändern wird oder nicht.

Zum Abschluß gibt es noch einige interessante Eigenschaften zu erwähnen: Bei erfolgreichem Austausch benötigt das Protokoll P5 lediglich zwei Nachrichten in den Modulen M2 und M3, was aufgrund der Wirksamkeitsbedingung offensichtlich optimal ist. Die Fehlerbehandlung könnte auch nicht einfacher sein, da nur die Partei A mit I5-opt-ww die Möglichkeit zur Konfliktlösung hat. Noch weniger würde gar keine Konfliktlösung bedeuten und damit wäre keine Fairneß mehr möglich. Daher handelt es sich bei Protokoll P5 um das einfachste optimistische Fairneßprotokoll.

Protokollvariationen. Im folgenden untersuche ich, wie sich Änderungen der Objekteigenschaften auf die Fairneßgarantien von P5 auswirken. Da nur eine Modulimplementierung zur Konfliktlösung zur Verfügung steht, müssen lediglich die folgenden beiden Fälle unterschieden werden: Entweder die Konfliktlösung durch Widerrufen von O_A scheitert oder das Widerrufen von O_A ist möglich.

Falls in I5-opt-ww das Objekt O_A nicht widerrufen werden kann, erzielt das Protokoll F0-Fairneß für A und F4-Fairneß für B. Dieser Fall tritt genau dann ein, wenn O_A oder O_B nicht widerrufbar ist.

Es bleibt der Fall, daß O_A und O_B mindestens schwach widerrufbar sind. Das Protokoll P5 liefert bei starker Widerrufbarkeit von O_A und mindestens schwacher Widerrufbarkeit von O_B F4-Fairneß für beide Parteien, wobei B jedoch mit T1-Terminierung der Form T1– vorlieb nehmen muß. Bei schwacher Widerrufbarkeit von O_A reduziert sich die Fairneß von A auf F2-Fairneß, weil das Widerrufen von O_A scheitern kann, wenn B dies verhindert. Da der Vermittler den Betrug von B bemerkt, ist eine Konfliktlösung durch ein Gericht außerhalb des Systems möglich.

Die Auswirkungen der verschiedenen Arten von Widerrufbarkeit auf die Fairneß von P5 sind in Tabelle 3.22 zusammengefaßt. Wenn nichts anderes angegeben ist, wird für beide Parteien T2-Terminierung erreicht.

3.1.7 Andere optimistische Protokolle

Alle bekannten optimistischen Fairneßprotokolle setzen Generierbarkeit oder Widerrufbarkeit in irgendeiner Form voraus (ausgenommen Protokolle mit Hardwareunterstützung wie in Kapitel 5). Ob sich optimistische Protokolle ohne diese beiden Eigenschaften finden lassen, ist ein offenes Problem und wurde bereits von Asokan [Aso98, S.25 bzw. S.137] aufgeworfen. Da nach dem heutigen Wissensstand optimistischen Protokolle nur mit Hilfe von Generierbarkeit oder Widerrufbarkeit F4-Fairneß erreichen können, beschränke ich die folgende Diskussion auf diese Eigenschaften.

Widerrufbarkeit		**Fairneß für**		**Anmerkung**
O_A	O_B	A	B	
stark	stark	F4	F4[1)]	[1)] nur T1-Terminierung für B
stark	schwach	F4	F4[1)]	[1)] nur T1-Terminierung für B
stark	–	F0	F4	
schwach	stark	F2	F4[1)]	Tausche A, $B \Rightarrow$ F4 für A. [1)] nur T1 für B
schwach	schwach	F2	F4[1)]	[1)] nur T1-Terminierung für B
schwach	–	F0	F4	
–	stark	F0	F4	
–	schwach	F0	F4	
–	–	F0	F4	

Tabelle 3.22: Einfluß der Widerrufbarkeit auf die Fairneß von Protokoll P5.

Bei den bisher vorgestellten optimistischen Protokollen müssen immer beide Objekte besondere Eigenschaften haben: Bei P2 bzw. P5 sind beide Objekte generierbar bzw. widerrufbar. Bei P3 und P4 ist jeweils ein Objekt generierbar und das andere widerrufbar. Bei den Protokollen P2 bis P5 wurden auch die verschiedenen Variationen von starker und schwacher Generierbarkeit bzw. Widerrufbarkeit diskutiert. Daher bleiben nur wenige mögliche Kombinationen übrig, die noch nicht untersucht wurden:

- Protokolle mit nur einem entweder generierbarem oder widerrufbarem Objekt.
- Protokolle mit einem generierbarem und widerrufbarem Objekt.

Im folgenden argumentiere ich dafür, daß in beiden Fällen kein Protokoll mit F4-Fairneß und T2-Terminierung konstruiert werden kann:

Im ersten Fall hat nur ein Objekt eine Eigenschaft, die zur Konfliktlösung verwendet werden kann. Aus diesem Grund kann nicht immer sichergestellt werden, daß der Vermittler die Konfliktlösung für beide Parteien erfolgreich durchführen kann. Im Abschnitt 3.1.3 kann der Vermittler zum Beispiel bei einem nur generierbaren Objekt nicht die T2-Terminierung für beide Parteien sicherstellen, weil sonst keine Fairneß mehr erreicht werden kann. Ebenso besteht in den Abschnitten 3.1.4, 3.1.5 und 3.1.6 das Problem, daß nur mit Widerrufbarkeit ebenso entweder die Fairneß oder die Terminierung abgeschwächt werden muß. Daher ist zur Zeit keine bessere Lösung bekannt, als bei einem stark generierbaren Objekt das Protokoll P2 zu verwenden und T0/T1-Terminierung in Kauf zu nehmen.

Im zweiten Fall stehen zwar aufgrund der Generierbarkeit und Widerrufbarkeit zwei verschiedene Ansätze zur Konfliktlösung zur Verfügung, aber sie wirken sich beide auf dasselbe Objekt aus. Wenn der Vermittler ein Objekt zuerst widerruft und dann später wieder generiert (oder umgekehrt), trifft er eine inkonsistente Entscheidung, die grundsätzlich verboten ist und außerdem immer die T2-Terminierung verletzen würde. Wenn er dagegen nur eine der beiden Eigenschaften zur Konfliktlösung ausnutzt, kann kein besseres Protokoll als im ersten Fall konstruiert werden.

Aus diesen Überlegungen folgt, daß die Protokolle P2 bis P5 vermutlich die einzigen optimistischen Protokolle sind, die einen für die Praxis hinreichend hohen Fairneßgrad erzielen. Falls die Vorbedingungen für diese optimistischen Protokolle nicht erfüllt sind, sollte daher immer auf das aktive Protokoll P1 ausgewichen werden.

3.2 Erweiterungen für Nichtabstreitbarkeit

Bei der Nichtabstreitbarkeit handelt es sich um eine optionale Protokolleigenschaft, auf die in vielen Fällen (z.B. Austausch signierter Verträge) verzichtet werden kann. In diesem Abschnitt zeige ich, wie Nichtabstreitbarkeit bei Bedarf als eine Erweiterung der bisher vorgestellten modularen Austauschprotokolle realisiert werden kann, ohne die wesentlichen Konzepte zu ändern. Damit liefert dieser Abschnitt die folgenden Ergebnisse:

- Anhand des Beispiels der Nichtabstreitbarkeit wird die besonders einfache Erweiterbarkeit der modularen Austauschprotokolle verdeutlicht.
- Die Erweiterung um Nichtabstreitbarkeit erleichtert den Vergleich mit anderen in der Literatur vorgeschlagenen Protokollen, die ebenfalls Nichtabstreitbarkeit gewährleisten.

Im folgenden stelle ich zuerst die Realisierung von Nichtabstreitbarkeit der Herkunft vor und diskutiere dann die nötigen Protokollerweiterungen für Nichtabstreitbarkeit des Empfangs.

3.2.1 Nichtabstreitbarkeit der Herkunft

Um den Nachweis zu erbringen, von wem ein Objekt stammt, kann der entsprechende Nachweis der Herkunft zusammen mit dem Objekt verschickt werden. Bei den Protokollen muß dann lediglich O_A durch $(O_A, \mathrm{NH}(A, O_A))$ ersetzt werden und ebenso O_B durch $(O_B, \mathrm{NH}(B, O_B))$. Dabei bezeichnet $\mathrm{NH}(P, O_P)$ den Nachweis dafür, daß die Partei P der Urheber des Objekts O_P ist. Der Empfänger eines so erweiterten Objekts prüft dann zusätzlich, ob der mitgelieferte Nachweis der Herkunft den Vereinbarungen entspricht.

Die Einzelheiten zur Implementierung dieses Konzepts werden in Abschnitt 4.4.3 diskutiert.

3.2.2 Nichtabstreitbarkeit des Empfangs

Um den Nachweis zu erbringen, daß eine Partei ein Objekt empfangen hat oder noch empfangen kann, müssen im wesentlichen die Implementierungen von Modul M3 erweitert werden. In diesem Modul müssen jetzt zusätzlich zu den Objekten auch noch die Nachweise des Empfangs ausgetauscht werden.

Implementierung

Beim aktiven Protokoll P1 kann ähnlich wie bei der Nichtabstreitbarkeit der Herkunft in Modul M3 das Objekt O_A durch (O_A, NE(V, O_B)) ersetzt werden und ebenso O_B durch (O_B, NE(V, O_A)). Dabei steht NE(V, O_P) für einen Nachweis des Empfangs und wird hier vom Vermittler V für das Objekt O_P geliefert. Dadurch können die Parteien A und B immer beweisen, daß die jeweils andere Partei das Objekt bekommen hat oder jederzeit vom Vermittler anfordern kann.

I3-opt-NE: Implementierung von Modul M3 für optimistischen Austausch		
A → B	:	O_A
B	:	Prüfe, ob O_A der Beschreibung entspricht
B → A	:	O_B, NE(B, O_A)
A	:	Prüfe, ob O_B und NE(B, O_A) der Beschreibung entsprechen
A → B	:	NE(A, O_B)
B	:	Prüfe, ob NE(A, O_B) der Beschreibung entspricht

Tabelle 3.23: Modul M3 für optimistisch fairen Austausch mit Nichtabstreitbarkeit des Empfangs.

I4-opt-NE: Implementierung von Modul M4 für optimistischen Austausch			
B → V	:	O_A, O_B, die Beschreibungen dieser beiden Objekte, NE(B, O_A)	
V	:	Falls O_A oder O_B schon vorher widerrufen wurde:	
		V → B	: Austausch bereits abgebrochen
		Falls ein Objekt oder NE(B, O_A) nicht der Beschreibung entspricht:	
		V → B	: Fehler: falsche Eingabewerte
		Sonst:	
		V	: Berechne NE(V, O_B)
			: Speichere O_B, NE(B, O_A)
		V → B	: NE(V, O_B)

Tabelle 3.24: Optimistisch fairer Austausch mit Nichtabstreitbarkeit des Empfangs. In dieser Implementierung von M4 möchte B den Nachweis des Empfangs für O_B vom Vermittler erhalten.

Bei den optimistischen Protokollen P2 bis P5 sind die verwendeten Implementierungen I3-opt-gg, I3-opt-gw und I3-opt-ww bis auf die Fehlerbehandlungen identisch. Daher genügt es, hier eine einzige Implementierung anzugeben, die in allen Fällen anwendbar ist. Die Implementierung I3-opt-NE in Tabelle 3.23 erweitert I3-opt-gg, I3-opt-gw und I3-opt-ww im wesentlichen um eine zusätzliche Nachricht. Wenn Partei B den Empfang von O_A mit NE(B, O_A) bestätigt hat, antwortet A mit der Empfangsbestätigung NE(A, O_B) für das Objekt O_B. Durch diese zusätzliche Nachricht entsteht ein neuer Fehlerfall, wenn B ein fehlerhaftes NE(A, O_B) oder einfach keine Antwort erhält. Zum Fortsetzen des Austausches kann B dann I4-opt-NE (siehe Tabelle 3.24) aufrufen.

Die Konfliktlösung durch I4-opt-NE setzt voraus, daß B bereits O_A erhalten hat. Dann schickt B die Beschreibungen der Objekte, sein eigenes Objekt O_B zusammen mit O_A und

der entsprechenden Empfangsbestätigung an den Vermittler. Wenn eines der Objekte widerrufbar ist und vom Vermittler bereits widerrufen wurde, dann teilt der Vermittler nur mit, daß der Austausch bereits abgebrochen wurde. Falls beide Objekte ihrer Beschreibung entsprechen und $\mathrm{NE}(B, O_A)$ eine gültige Empfangsbestätigung für O_A ist, wird der Vermittler den Austausch fortsetzen. Er berechnet eine Ersatzempfangsbestätigung $\mathrm{NE}(V, O_B)$ für das Objekt O_B, speichert O_B und $\mathrm{NE}(B, O_A)$ für mögliche zukünftige Anfragen von A ab und liefert $\mathrm{NE}(V, O_B)$ an B.

Falls nach der Ausführung von I4-opt-NE A eine Konfliktlösung startet, wird der Vermittler den Austausch immer fortsetzen. Zusätzlich zum Objekt O_B erhält A dann auch die Empfangsbestätigung $\mathrm{NE}(B, O_A)$.

Ein Sonderfall tritt ein, wenn zwei generierbare Objekte ausgetauscht werden und A vor der Ausführung von I4-opt-NE den Austausch mit I5-opt-gg abgebrochen hat. Dann wird B mit I4-opt-NE trotzdem den Austausch beenden können. Dies ist kein Widerspruch, da B durch den Besitz von O_A beweisen kann, daß A den Austausch trotz des Abbruchs fortgesetzt hat. Dies setzt allerdings voraus, daß B einen zu O_A passenden Nachweis der Herkunft besitzt. Weitere Informationen zu diesem Sonderfall werden in [BK00] beschrieben.

Problematisch ist die Erweiterung um einen Nachweis des Empfangs nur bei Protokoll P5 mit zwei widerrufbaren Objekten. Bei diesem Protokoll besteht die Gefahr, daß das Objekt O_A in der Zwischenzeit widerrufen wurden, wodurch auch die entsprechende Empfangsbestätigung $\mathrm{NE}(B, O_A)$ ungültig wird. Daher müßte man in P5 immer beim Vermittler nachfragen, ob die Empfangsbestätigung $\mathrm{NE}(B, O_A)$ gültig ist. Selbst wenn der Vermittler dies bestätigt, besteht wegen der T1-Terminierung immer noch das Problem, daß das Objekt im nächsten Augenblick von A widerrufen werden kann. Aus diesem Grund sollte beim Protokoll P5 der Nachweis des Empfangs vom Typ $\mathrm{NE}(B, O_A)$ immer ignoriert werden. Dagegen ist die Empfangsbestätigung $\mathrm{NE}(A, O_B)$ auch bei P5 sinnvoll, wie das ausführliche Beispiel am Ende dieses Abschnitts zeigt.

In allen anderen Fällen gibt es keine Konflikte zwischen der Widerrufbarkeit von Objekten und den Nachweisen des Empfangs.

Nachweis der Protokolleigenschaften

Um die Sicherheit der durch die neuen Implementierungen ergänzten Protokolle zu zeigen, müßten genaugenommen für jedes Protokoll neue Beweise geführt werden. Da sich die Eigenschaften der ergänzten Protokolle jedoch nicht wesentlich ändern, argumentiere ich hier nur, ob es Auswirkungen gibt und wenn ja welche.

Wirksamkeit: Die neue Implementierung I3-opt-NE sorgt bei korrekter Ausführung für den Austausch der Objekte und der Nachweise des Empfangs.

Terminierung: In I3-opt-NE gibt es einen zusätzlichen Fehlerfall, der die Terminierung beeinflußt. Wenn B keine Empfangsbestätigung erhält, kann er nicht einfach das Protokoll beenden, sondern muß I4-opt-NE aufrufen. Diese Implementierung liefert bei korrekter Eingabe immer eine Antwort, die zur Terminierung von B führt.

In allen anderen Fällen wird die Terminierung durch die Erweiterungen für die Nichtabstreitbarkeit nicht beeinflußt.

Fairneß: Der Austausch der Objekte wird von der neuen Implementierung kaum beeinflußt. Nur in I4-opt-NE speichert der Vermittler vor der Lieferung des Nachweis des Empfangs das Objekt O_B und die Empfangsbestätigung $\mathrm{NE}(B, O_A)$. Daher werden alle folgenden Anfragen an den Vermittler immer zur erfolgreichen Weiterführung des Austausches führen und somit Fairneß gewährleisten. Aus diesem Grund können die Protokolländerungen zu keinem Verlust der Fairneß führen.

Nichtabstreitbarkeit: Beweise für die Nichtabstreitbarkeit der Herkunft werden von den Änderungen nicht beeinflußt.
Die Nichtabstreitbarkeit des Empfangs wird durch die vorgeschlagenen Änderungen für die Parteien A und B sichergestellt. Die einzige Ausnahme besteht in der Empfangsbestätigung $\mathrm{NE}(B, O_A)$ bei Protokoll P5. Diese kann nicht von A verwendet werden, da A das dazugehörige O_A immer noch widerrufen kann. Somit garantiert das erweiterte Protokoll P5 nur Nichtabstreitbarkeit des Empfangs für Partei B.

Beispiele

Die Protokolle P2 bis P5 lassen sich leicht anpassen, so daß sie Nichtabstreitbarkeit des Empfangs gewährleisten. Der wesentliche Unterschied besteht in der zusätzlichen Implementierung I4-opt-NE. Die erweiterten Protokolle sehen dann wie folgt aus:

P2-NE: I1, I2-opt-gg, I3-opt-NE, I4-opt-gg, I4-opt-NE, I5-opt-gg.

P2'-NE: I1, I2-opt-gg, I3-opt-NE, I4-opt-gg, I4-opt-gg', I4-opt-NE, I5-opt-gg.

P3-NE: I1, I2-opt-gw, I3-opt-NE, I4-opt-gw, I4-opt-NE, I5-opt-gw.

P4-NE: I1, I2-opt-gw, I3-opt-NE, I4-opt-gw', I4-opt-NE, I5-opt-gw.

P5-NE: I1, I3-opt-NE, I4-opt-NE, I5-opt-ww.

Besonders erwähnenswert ist hier das Protokoll P5-NE. In Abschnitt 3.1.6 wurde gezeigt, daß bei zwei widerrufbaren Objekten für die Partei B nur T1-Terminierung erreicht wird. Deshalb kann diese Partei nicht immer sicher sein, daß ein erfolgreicher Austausch nicht doch noch zurückgesetzt wird. Der Einsatz einer Empfangsbestätigung kann hier mehr Klarheit schaffen. Sobald B in I3-opt-NE die Empfangsbestätigung für O_B erhalten hat, kann er davon ausgehen, daß A sein Objekt nicht widerrufen wird, weil er nachweislich im Besitz von O_B ist.

Die Empfangsbestätigung gibt B also einen deutlichen Hinweis, ob sich der Zustand des Austausches noch ändern wird. B kann damit besser einschätzen, welches Ergebnis das Protokoll mit T1-Terminierung haben wird. Die Empfangsbestätigung kann allerdings nicht verhindern, daß A trotz ausgestellter Empfangsbestätigung den Widerruf der Objekte beim Vermittler mit I5-opt-ww veranlaßt und dabei gegenüber dem Vermittler behauptet, er habe O_B nicht erhalten. Dann kann B jedoch, sobald er vom Widerruf

beider Objekte erfährt, die Empfangsbestätigung von A beim Vermittler vorlegen und so den Betrug von A entlarven.

Angenommen, ein solcher nachgewiesener Betrug zieht ernsthafte Konsequenzen für A nach sich. Da B das Widerrufen von O_A und O_B in den meisten Fällen bemerken wird (z.B. weil das verwendete Zahlungssystem den Widerruf der Zahlung meldet), geht A ein hohes Risiko ein, wenn er eine Empfangsbestätigung versendet und trotzdem widerruft. Deshalb kann man in der Praxis davon ausgehen, daß dieser Fall fast nie vorkommen wird, so daß das Protokoll P5-NE im Prinzip auch T2-Terminierung für B sicherstellt.

3.3 Zusammenfassung

In diesem Kapitel habe ich gezeigt, wie basierend auf der in Abschnitt 2.3.3 eingeführten Modellierung eine einheitliche Beschreibung von aktiven und optimistischen Fairneßprotokollen verwirklicht werden kann. Daher kann der Fairneßdienst FlexiFair eine Vielzahl verschiedener Protokolle bereitstellen, die sich alle über die gleiche Schnittstelle ansprechen lassen. Eine Anwendung muß sich somit nicht danach richten, wie ein Fairneßprotokoll implementiert ist. Es muß bei der Auswahl eines Protokolls lediglich darauf geachtet werden, daß die ausgetauschten Objekte die Vorbedingungen für das entsprechende Protokoll erfüllen und daß das Protokoll dann die gewünschten Fairneßeigenschaften garantiert.

Durch die systematische Untersuchung von optimistischen Fairneßprotokollen in Abschnitt 3.1 habe ich mehrere neue Fairneßprotokolle entwickelt, die speziell die Objekteigenschaft Widerrufbarkeit ausnutzen. Insbesondere die Protokolle P3 und P4 stellen einen deutlichen Fortschritt dar, weil sie mit Widerrufbarkeit und Generierbarkeit die gleichen Protokolleigenschaften realisieren können, wie das nur auf Generierbarkeit basierende Protokoll P2. Das Protokoll P5 schließlich zeigt, daß auch ganz ohne Generierbarkeit fairer Austausch möglich ist.

Alle Protokolle lassen sich mit den Maßnahmen aus Abschnitt 3.2 leicht erweitern, so daß sie Nichtabstreitbarkeit der Herkunft und des Empfangs gewährleisten. Im Fall von Protokoll P5-NE führt diese Erweiterung sogar zu einer Verbesserung der Konfliktlösung, da eine benachteiligte Partei jetzt einen Nachweis des Empfangs vorlegen kann.

Bei der Beschreibung der optimistischen Protokolle profitiere ich von den in Abschnitt 2.3.4 neu eingeführten Begriffen der starken und schwachen Generierbarkeit bzw. Widerrufbarkeit. Diese Begriffe ermöglichen die generische Beschreibung von Protokollen, ohne daß diese Abstraktion zu Einschränkungen führt: Die so modellierten Objekteigenschaften sind hinreichend für die Konstruktion von optimistischen Protokollen. Es ist sogar möglich, Protokollvariationen zu erzeugen, indem man Objekte mit schwächeren Eigenschaften verwendet, was meist zu reduzierten Fairneßeigenschaften führt. Durch diese Variationen lassen sich also vorhandene Protokolle auf unterschiedliche Weisen nutzen. In Tabelle 3.25 sind die Auswirkungen von verschieden starker Generierbarkeit und Widerrufbarkeit auf optimistische Protokolle zusammengefaßt. Anhand dieser Tabelle läßt sich zu gegebenen Objekteigenschaften leicht ein passendes Protokoll finden. Falls die in der Tabelle angegebenen Fairneßgarantien zu gering sein sollten, kann immer noch auf

Protokoll P1 zurückgegriffen werden, das für beliebige digitale Objekte F4-Fairneß mit T2-Terminierung garantiert. Auf diese Weise kann jetzt eine Anwendung automatisch entscheiden, welches der von FlexiFair angebotenen Protokollen eingesetzt werden soll.

Eigenschaft		**Fairneß für**		**Anmerkung**
O_A	O_B	A	B	
G	G	F4	F4	Protokoll P2
g	G	F4	F4	Protokoll P2'
W	G	F4	F4	Protokoll P3 oder P4
w	G	F4	F4	Protokoll P4
–	G	F4	F4[1)]	Protokoll P2, [1)] T0/T1-Terminierung für B
g	g	F2	F4	Protokoll P2'
W	g	F4	F4	Protokoll P3
w	g	F2	F4	Protokoll P3 oder P4
–	g	F2	F4[1)]	Protokoll P2, [1)] T0/T1-Terminierung für B
W	W	F4	F4[2)]	Protokoll P5, [2)] T1– für B, daher besser P5-NE
w	W	F4[2)]	F4	Protokoll P5, [2)] T1– für A, daher besser P5-NE
–	W	F0	F4	Protokoll P3 oder P5
w	w	F2	F4[2)]	Protokoll P5, [2)] T1– für B, daher besser P5-NE
–	w	F0	F4	Protokoll P5
–	–	F0	F4	Protokoll P2–P5
–	GW	F4	F4[1)]	Protokoll P2, [1)] T0/T1-Terminierung für B
–	Gw	F4	F4[1)]	Protokoll P2, [1)] T0/T1-Terminierung für B
–	gW	F2	F4[1)]	Protokoll P2, [1)] T0/T1-Terminierung für B
–	gw	F2	F4[1)]	Protokoll P2, [1)] T0/T1-Terminierung für B

G : stark generierbar W : stark widerrufbar
g : schwach generierbar w : schwach widerrufbar

Tabelle 3.25: Einfluß der Widerrufbarkeit und Generierbarkeit auf die Fairneß von optimistischen Protokollen.

4 Implementierung

Der von mir entwickelte Fairneßdienst FlexiFair soll unterschiedlichen Anwendungen den Einsatz von fairem Austausch ermöglichen. Mit der in Abschnitt 2.3.3 beschriebenen Modellierung von Austauschprotokollen biete ich eine einheitliche Schnittstelle für Austauschprotokolle, die von den in Kapitel 3 beschriebenen Protokollen implementiert wird. Zusätzlich zu den Protokollen stellt FlexiFair auch die Implementierung für Objekte und ihre Beschreibungen zur Verfügung. Auch dabei wird das Ziel verfolgt, eine möglichst vielseitige Implementierung zu entwickeln, die von vielen Anwendungen genutzt werden kann.

In diesem Kapitel wird die Implementierung von FlexiFair vorgestellt. Sie wurde in der Programmiersprache Java durchgeführt. Die Wahl von Java hatte mehrere Gründe: Erstens stellt Java eine flexible Kryptoarchitektur zur Verfügung stellt, zweitens ermöglicht es das Verschicken von ausführbaren Programmen mittels Serialisierung von Java-Objekten und drittens können die Ausführungsrechte von Klassen beschränkt werden, um die Sicherheit zu erhöhen.

Die prototypische Implementierung von FlexiFair wirft die folgenden Probleme auf, die in diesem Kapitel behandelt werden:

Sicherheit: Gesendete Nachrichten müssen sich einer bestimmten Austauschtransaktion zuordnen lassen. Das Wiedereinspielen oder Wiederverwenden von alten Nachrichten darf nicht möglich sein. Außerdem müssen sich die beteiligten Parteien authentisieren, damit unbefugte Parteien keinen Einfluß auf den Austauschvorgang nehmen können.

Erweiterbarkeit: Neue Anwendungen sollen den Fairneßdienst ohne langwierige Anmeldung nutzen können. Auch wenn eine Anwendung eigene Objekte definiert, die sie austauschen will, sollte ein Austausch ohne Änderungen am Vermittler möglich sein.

Einsatz in speziellen Austauschszenarien: Die Austauschprotokolle sind bewußt generisch gehalten, so daß viele Anwendungen sie nutzen können. Daher muß gezeigt werden, wie Anwendungen die geeigneten Protokolle, digitalen Objekte und ihre Beschreibungen für z.B. Vertragsunterzeichnung, E-Mail-Empfangsbestätigung oder Einkauf von Waren auswählen und welche Objekte und Beschreibungen für diese Szenarien geeignet sind.

Durch die Lösung dieser Probleme wird die Praktikabilität der Konzepte für den Fairneßdienst FlexiFair verdeutlicht. Bei der Implementierung muß keineswegs auf die im Entwurf von FlexiFair enthaltene Flexibilität verzichtet werden.

Im folgenden werden zuerst allgemeine Implementierungsfragen in den Abschnitten 4.1 und 4.2 diskutiert. Beinahe alle Veröffentlichungen (Ausnahmen bilden z.B. [LNJ00, Lou00]) ignorieren diese Implementierungsdetails und liefern stattdessen nur sehr abstrakte Protokollbeschreibungen. Dabei sind besonders die Sicherheitsfragen hervorzuheben, die in vielen anderen Veröffentlichungen ignoriert werden (z.B. [BP90, BF98, VPG99, FPH00]) oder unzureichend gelöst sind (z.B. in [ZDB00, KM00, BK00, MK01] wurden verschiedene Angriffe übersehen). In Abschnitt 4.3 wird die Realisierung von Objektbeschreibungen erläutert, wobei der dort vorgeschlagene dynamische Ansatz die Funktionalität von FlexiFair zur Laufzeit dynamisch erweitern kann. Die Implementierung der Objekte sowie ihrer besonderen Eigenschaften wird in Abschnitt 4.4 vorgestellt. In Abschnitt 4.5 wird die Implementierung der Austauschprotokolle beschrieben und in Abschnitt 4.6 folgen die Details zur Realisierung des Vermittlers.

Am Ende des Kapitels gebe ich anhand der Beispielszenarien Vertragsunterzeichnung, E-Mail-Empfangsbestätigung, Einkauf einer Ware und Aufdecken von Pseudonymen einen Ausblick über die vielfältigen Einsatzmöglichkeiten von FlexiFair. Es wird gezeigt, wie für diese Szenarien ein Protokoll und die geeigneten Objekte bzw. Beschreibungen ausgewählt werden können. Beim Szenario des simultanen Aufdeckens von Pseudonymen handelt es sich um ein von mir neu eingeführtes Austauschszenario, das in der Literatur bisher noch nicht untersucht wurde. Trotzdem kann FlexiFair eine Lösung für die Realisierung von fairem Austausch in diesem Szenario anbieten, was die Vielseitigkeit von FlexiFair deutlich unterstreicht.

4.1 Sicherheit auf Implementierungsebene

Bei der bisherigen Beschreibung des Fairneßdienstes FlexiFair wurden viele Details ausgeklammert. Die Fairneßprotokolle aus Kapitel 3 sind zum Beispiel bewußt sehr abstrakt aufgeschrieben, um deren Protokolleigenschaften zu verdeutlichen. Bei der konkreten Implementierung dieser Protokolle werden jedoch die in diesem Abschnitt eingeführten, zusätzlichen Sicherheitsmaßnahmen benötigt.

In vielen Veröffentlichungen sind die Sicherheitsmaßnahmen zu knapp bemessen oder sie werden fehlerhaft umgesetzt, so daß Angriffe gegen diese Protokolle möglich sind. Die folgenden Beispiele mit bisher unveröffentlichten Angriffen illustrieren solche Sicherheitsprobleme:

- In [KM00, MK00, MK01, MS01, KMZ02] kann das für den Vermittler verschlüsselte Objekt der Partei A von der anderen Partei B wiederverwendet werden. In einer neuen Transaktion TID' entschlüsselt der Vermittler dann das Objekt von A, falls B dabei einen Austausch mit einer Partei A' vortäuscht. Durch diesen Implementierungsfehler geht die Fairneß für die Partei A verloren.

- In [ZDB99, ZDB00, BK00] schickt A das für den Vermittler verschlüsselte Objekt O_A unsigniert an B. Falls A einen falschen Wert schickt, kann B nicht beweisen, daß dieser Chiffretext wirklich von A stammt. Daher kann der Vermittler keine Kon-

fliktlösung für B durchführen, so daß das Protokoll die Terminierungseigenschaft für B verliert.

Durch die in diesem Abschnitt vorgestellten Sicherheitsmaßnahmen sollen solche Angriffe verhindert werden. Das Ziel der hier vorgeschlagenen Maßnahmen ist es, generische Sicherheitskonzepte für beliebige Austauschprotokolle und beliebige ausgetauschte Objekte zu erhalten. Eine sichere Implementierung sollte mindestens die folgenden Anforderungen erfüllen:

- Es gibt eine global eindeutige Bezeichnung der aktuellen Transaktion, die die zuverlässige Unterscheidung verschiedener Austauschtransaktionen erlaubt.
- Es muß festgelegt sein, wer die Rollen von Partei A und B übernimmt. Keine andere Partei darf sich als A bzw. B ausgeben können.
- Die Authentizität und Integrität der ausgetauschten Nachrichten muß überprüfbar sein.
- Ein Wiedereinspielen und Wiederverwenden von alten Nachrichten bzw. Nachrichtenteilen muß erkannt werden.

Die ersten beiden Punkte können mit der im folgenden Abschnitt beschriebenen eindeutigen Transaktionsnummer gelöst werden. Für die letzten beiden Punkte ist der richtige Einsatz von Signaturen und Verschlüsselung notwendig, was anschließend diskutiert wird.

4.1.1 Transaktionsnummern und Authentisierung der Parteien

Die Parteien sowie der Vermittler benötigen einen global eindeutigen Bezeichner, der die zuverlässige Unterscheidung verschiedener Austauschtransaktionen erlaubt. Daher muß eine sogenannte *Transaktionsnummer* TID für jeden Austauschvorgang gewählt werden. Dabei muß darauf geachtet werden, daß diese Nummer bei jedem Austausch verschieden ist und daß ein Angreifer nicht willkürlich mehrmals die gleiche Nummer wählen kann.

Die Grundidee bei der Wahl der Transaktionsnummer besteht in der Berechnung mittels einer Hashfunktion. Wenn beide Parteien eine Zahl $Rand_A$ bzw. $Rand_B$ wählen, aus der $TID = H(Rand_A, Rand_B)$ berechnet wird, dann hat jede Partei Einfluß auf die richtige Wahl der Transaktionsnummer. Da eine Hashfunktion praktisch kollisionsfrei ist, kann eine Partei das Auftreten einer bereits verwendeten Transaktionsnummer dadurch verhindern, daß sie ihre Zahlen immer verschieden wählt. In der Praxis kann man dazu z.B. große Zufallszahlen oder die lokale Systemzeit in Millisekunden verwenden.

Diese Realisierung stellt jedoch nur sicher, daß eine Partei nie zwei mal die gleiche Transaktionsnummer verwendet. Ein Angreifer könnte in einer anderen Austauschtransaktion die gleiche Transaktionsnummer verwenden, um die Ausführung des urspünglichen Austausches zu behindern oder zu beeinflussen. Aus diesem Grund muß die Transaktionsnummer auch mit der Authentisierung der Parteien verknüpft werden. Dies sollte durch das Hashen der zur Authentisierung verwendeten öffentlichen Signaturschlüssel $PubKey_A$

und $PubKey_B$ erfolgen, so daß $TID = H(PubKey_A, PubKey_B, Rand_A, Rand_B)$ berechnet wird.

Ein von A und B verschiedener Angreifer kann jetzt den Austausch nicht mehr stören, da er keine mit den in TID enthaltenen Schlüsseln überprüfbare Signaturen ausstellen kann, die die Authentizität und Integrität von Nachrichten für diese Transaktion beweisen würden. Diese Wahl der Transaktionsnummer hat außerdem den Vorteil, daß der Vermittler nach einer Authentisierung mit dem entsprechenden privaten Schlüssel genau weiß, welche Rolle diese Partei bei dem Austausch spielt. Ein Vertauschen von A und B würde nämlich immer zu einer anderen Transaktionsnummer führen.

Damit sich bei einem späteren Disput nachweisen läßt, welches Austauschprotokoll mit welchem Vermittler verwendet wurde, muß immer ein eindeutiger Bezeichner für das verwendete Protokoll und den möglicherweise beteiligten Vermittler V in der Transaktionsnummer enthalten sein. Das führt dann zu folgender Transaktionsnummer:

$$TID = H(ProtID, V, PubKey_A, PubKey_B, Rand_A, Rand_B)$$

Der Bezeichner $ProtID$ steht dabei für das verwendete Protokoll zusammen mit den entsprechenden Parametern (z.B. ob das Protokoll mit oder ohne Nichtabstreitbarkeit des Empfangs arbeiten soll). Der Bezeichner für den Vermittler V kann in der Implementierung z.B. mit dem eindeutigen Namen des Vermittlers, seiner Internetadresse oder einfach seinem öffentlichen Schlüssel $PubKey_V$ realisiert werden. Auf diese Weise wird der Angriff ausgeschlossen, daß eine Partei verschiedene Vermittler zur Konfliktlösung wählt, so daß der Austausch bei dem einen abgebrochen und bei dem anderen fortgesetzt werden kann.

Eine weitere Verbesserung der Sicherheit erreicht man durch die zusätzliche Verknüpfung der Transaktionsnummer mit den Beschreibungen der auszutauschenden Objekte, die hier $desc_{O_A}$ und $desc_{O_B}$ genannt werden. Wenn die Transaktionsnummer auf

$$TID = H(ProtID, V, PubKey_A, PubKey_B, Rand_A, Rand_B, desc_{O_A}, desc_{O_B})$$

gesetzt wird, kann keine Partei später behaupten, daß es sich bei dieser Transaktion um den Austausch von anderen Objekten gehandelt hat.

Um einen Hashwert von der Beschreibung anzulegen, muß eine eindeutige Darstellung dieser Beschreibung existieren. In Abschnitt 4.3 wird gezeigt, daß die Beschreibung sowohl Daten als auch ausführbaren Programmcode umfassen sollte und durch ein Java-Objekt realisiert werden kann. Daher berechne ich den Hashwert von diesem Java-Objekt, indem ich es mittels Java-Serialisierung in einen Bitstring konvertiere. Der Programmcode der für die Beschreibung verwendeten Klasse wird im serialisierten Bitstring durch den vollständigen Klassennamen und den eindeutigen Wert der Klassenvariable `serialVersionUID` repräsentiert. Die im Java-Objekt enthaltenen Daten werden bei der Serialisierung automatisch gespeichert, so daß der serialisierte Bitstring die Daten und den Programmcode der Beschreibung enthält. Die genaue Art und Weise, wie ein Objekt serialisiert wird, läßt sich in Java sogar beim Programmieren der Klasse nach den eigenen Vorstellungen neu definieren.

4.1.2 Eindeutige Zuordnung von Nachrichten

Die Transaktionsnummern werden in FlexiFair verwendet, um Nachrichten eindeutig einer bestimmten Transaktion zuzuordnen. Die Verknüpfung einer Nachricht m mit der Transaktionsnummer TID erfolgt dadurch, daß (TID, m) mit dem Schlüssel signiert wird, dessen öffentlicher Schlüssel vom Absender zur Berechnung von TID gewählt wurde. Wenn zum Beispiel die Partei A in einem Austauschprotokoll mit der Transaktionsnummer TID die Nachricht m an Partei B schickt, so muß eine von A signierte Nachricht $s = Sig_A(TID, m)$ verwendet werden, um eine eindeutige Zuordnung zu erreichen. Der Empfänger B kann diese Signatur mit dem öffentlichen Schlüssel $PubKey_A$ verifizieren und dadurch beweisen, daß er diese signierte Nachricht in dem Austauschprotokoll mit der Transaktionsnummer TID von A bekommen hat. Damit eine solche Signatur nicht falsch interpretiert werden kann, sollte der Text der Nachricht auch die genaue Funktion dieser Nachricht im Austauschprotokoll angeben. Auf diese Weise kann das Wiedereinspielen von alten Protokollnachrichten an anderer Stelle im Protokoll erkannt werden.

Das Unterschreiben von Nachrichten schützt jedoch nicht vor Angriffen, bei denen die ursprüngliche Nachricht m von einem Angreifer C in einer neuen signierten Nachricht $\overline{s} = Sig_C(\overline{TID}, m)$ für ein anderes Austauschprotokoll mit der Transaktionsnummer $\overline{TID}$ wiederverwendet wird. Als Gegenmaßnahme schlage ich die Verschlüsselung der Nachricht für den beabsichtigten Empfänger vor, was zu einer Nachricht der Form

$$s = Sig_A(TID, E_B(TID, m)) \tag{4.1}$$

führen kann, bei der $E_B(TID, m)$ ein mit dem öffentlichen Schlüssel von B verschlüsselter Chiffretext ist. Der Empfänger B wird diese Nachricht nur dann akzeptieren, wenn sie beim Entschlüsseln die richtige Transaktionsnummer liefert. Daher kann der verschlüsselte Teil der Nachricht nicht einfach in einer Nachricht einer anderen Transaktion wiederverwendet werden. Insbesondere bei generierbaren und widerrufbaren Objekten muß auf die richtige Verknüpfung von Nachrichten mit der Transaktionsnummer geachtet werden, was auch in Abschnitt 4.4.1 anhand von weiteren Beispielen beschrieben wird.

Auch innerhalb einer Transaktion muß sich jede Nachricht eindeutig zuordnen lassen, um das Wiederverwenden einer Nachricht an anderen Stellen des Austauschprotokolls zu verhindern. Bei der in Formel (4.1) angegebenen Nachricht kann zum Beispiel eine falsche Interpretation nicht ausgeschlossen werden. Daher sollten Nachrichten zusätzlich zur Transaktionsnummer TID auch noch einen Bezeichner enthalten, der die genaue Funktion dieser Nachricht beschreibt. Wenn zum Beispiel Partei A einen Nachweis der Herkunft für sein Objekt liefert, dann sollte diese Nachricht die Form $Sig_A(TID, \mathrm{NH}, O_A)$ besitzen. Durch den Bezeichner NH weiß dann ein Empfänger, das in dieser Nachricht O_A signiert sein muß.

4.1.3 Diskussion

Die Transaktionsnummer TID stellt eine Verknüpfung mit *allen* für den Austausch wesentlichen Informationen her. Sobald sich nur einer dieser Austauschparameter ändert,

wird sich auch die Transaktionsnummer ändern. Dadurch werden viele Angriffe verhindert, weil jede Partei die Berechnung der Transaktionsnummer selbst durchführen kann und so jeden Fehler entdeckt.

Wenn eine Nachricht die Transaktionsnummer TID enthält und von einer in TID genannten Partei signiert ist, kann die Gültigkeit dieser Nachricht leicht überprüft werden. Ein Beispiel für eine solche Nachricht ist in Abbildung 4.1 dargestellt. Um Nichtabstreitbarkeit des Empfangs zu gewährleisten (siehe auch 4.4.4), signiert der Empfänger A das Objekt von B, was $Sig_A(TID, \mathrm{NE}, O_B)$ ergibt. Der für die Funktion der Signatur verwendete Bezeichner NE sagt aus, daß dies ein Nachweis des Empfangs sein soll. Auf diese Weise läßt sich die Aussage dieser Nachricht und ihr Zusammenhang mit den beim Austausch verwendeten Protokollparametern problemlos nachprüfen: Es genügt, wenn die Verifikation dieser Signatur mit dem in TID referenzierten öffentlichen Schlüssel $PubKey_A$ erfolgreich verläuft.

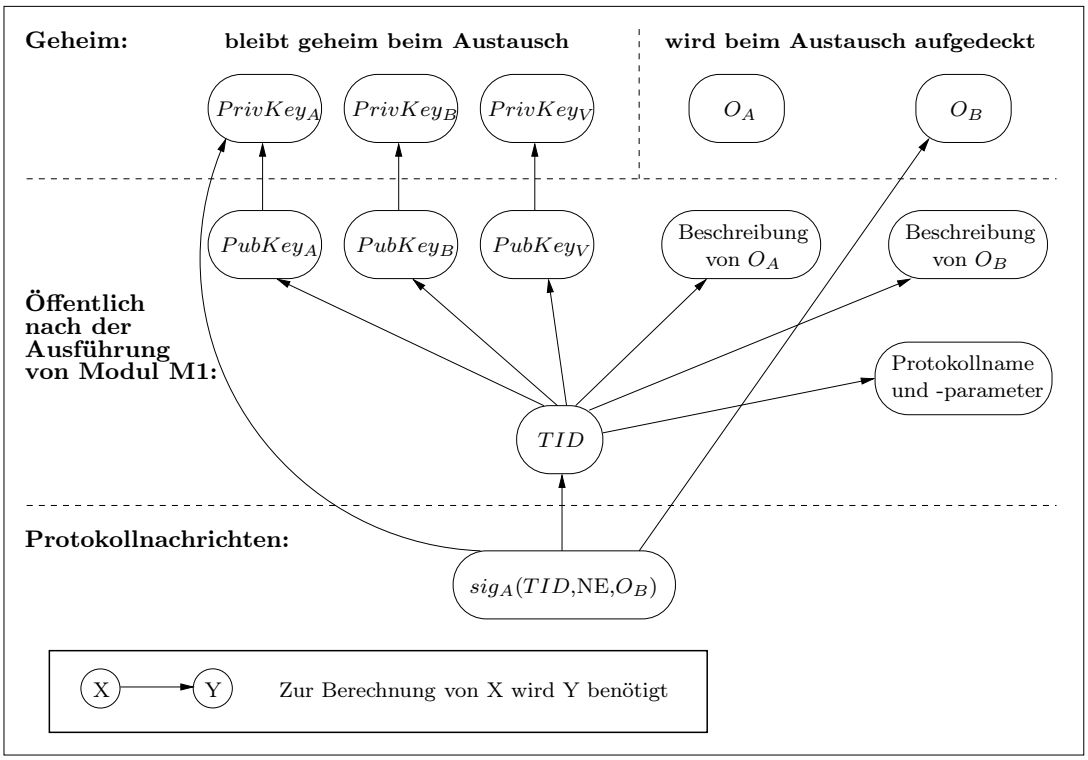

Abbildung 4.1: Die Transaktionsnummer TID stellt die Verbindung zwischen den bekannten Informationen und Nachrichten her. Ein Beispiel für eine Protokollnachricht ist eine Signatur als Nachweis des Empfangs, was durch NE gekennzeichnet ist (siehe auch Abschnitt 4.4.4).

Die hier vorgeschlagenen Maßnahmen zur Sicherung der Nachrichten von Austauschprotokollen sind relativ umfangreich. Der Grund dafür besteht darin, daß diese Maßnah-

men auf beliebige Protokolle, Objekte und Beschreibungen anwendbar sein müssen. In manchen Szenarien werden dadurch Angriffe verhindert, die sowieso nicht durchführbar wären. Ein Zuviel an Sicherheit schadet jedoch nicht. Dagegen haben die am Anfang dieses Abschnitts 4.1 genannten Beispiele verdeutlicht, daß die in vielen Veröffentlichungen beschriebenen Sicherheitsmaßnahmen zu knapp bemessen sind, wodurch verschiedene Angriffe möglich sind. Durch die in diesem Abschnitt vorgestellten Sicherheitsmaßnahmen hätten diese Angriffe jedoch leicht verhindert werden können.

4.2 Allgemeine Implementierungsprobleme

In diesem Abschnitt beschäftige ich mich mit verschiedenen Implementierungsproblemen. Als erstes wird die Speicherung des Protokollzustandes beschrieben, dann in Abschnitt 4.2.2 die Realisierung der Kommunikation und am Ende der Einfluß von Zeitbegrenzungen. Gute Konzepte zur Lösung dieser Implementierungsprobleme sind essentiell, um einen Fairneßdienst wie FlexiFair möglichst fehlerfrei zu realisieren.

4.2.1 Persistenz des Protokollzustands

Ein Austausch sollte sich auch nach einem Absturz der Anwendung bzw. des ganzen Rechners zu einem fairen Abschluß bringen lassen. Daher muß während der Ausführung eines Austauschprotokolls vor allem bei kritischen Operationen der Programmzustand persistent gesichert werden [LNJ00]. Es genügt bereits, wenn direkt vor jeder gesendeten Nachricht der Zustand gespeichert wird. Nach dem Zurücklesen dieses Zustands muß dann so fortgesetzt werde, als ob die Nachricht erfolgreich gesendet wurde. Dies ist nötig, da man bei einem Fehler beim Senden kaum entscheiden kann, ob die Nachricht angekommen ist. Daher ist es sicherer, wenn man annimmt, daß die Nachricht doch den Empfänger erreicht hat.

Beim Neustart einer Anwendung muß dann immer geprüft werden, ob ein gespeicherter Zustand für diese Austauschtransaktion existiert. Wenn dies der Fall ist, wird dieser Zustand beim Initialisieren des Austauschprotokolls eingelesen. Dann kann der Austausch fortgesetzt werden, indem eine geeignete Konfliktlösung gestartet wird. Daß eine solche Konfliktlösung immer vorbereitet ist, wird durch die Fairneßeigenschaft der Protokolle garantiert. Für die Protokolle macht es nämlich keinen Unterschied, ob sie wegen einem Betrugsversuch den Austausch nicht beenden können oder wegen einem Programmabsturz. Daher kann in beiden Fällen dieselbe Konfliktlösung durchgeführt werden.

Der Vermittler nimmt die persistente Speicherung des Zustands in seiner Datenbank vor, so daß er nach einem Neustart immer auf alle früheren Austauschvorgänge zugreifen kann. Die Anwendungen, die auf den Fairneßdienst FlexiFair zugreifen, werden dagegen den Zustand eher direkt auf der Festplatte speichern. Aus diesem Grund gibt es bei der Implementierung von FlexiFair verschiedene Klassen vom Typ `Storage` für die Zustandsspeicherung. `DBStorage` erlaubt die Zustandsspeicherung in einer Datenbank, während `FileStorage` den Zustand in eine Datei sichert.

4.2.2 Kommunikation

Die Austauschprotokolle übernehmen bei FlexiFair selbst die Abwicklung der Kommunikation zwischen verschiedenen Parteien. Um trotzdem unterschiedliche Wege der Datenübertragung in Austauschprotokollen zu unterstützen, wurde das Interface `Connection` definiert. Dieses Interface definiert verschiedene Methoden, um Java-Objekte zu versenden und zu empfangen. Das Austauschprotokoll wickelt daher seine gesamte Kommunikation über Implementierungen dieses Interfaces ab. Auf diese Weise bleiben die Eigenheiten der Kommunikation verborgen. Zum Beispiel öffnet die Klasse `ServerConnection` eine Verbindung erst, wenn das erste Mal Daten übertragen werden sollen. Im Gegensatz dazu arbeitet die Klasse `StreamConnection` nur mit einer bereits hergestellten Verbindung. Es wird also davon abstrahiert, ob es sich bei `Connection` um verbindungslose oder verbindungsorientierte Kommunikation handelt.

Ein weiterer Vorteil dieser Abstraktion der Kommunikation besteht darin, daß sich ein Austauschprotokoll nicht um die Verschlüsselung der Kommunikation kümmern muß. Falls die Daten verschlüsselt ausgetauscht werden sollen, muß die Anwendung dem Austauschprotokoll einfach eine Implementierung von `Connection` übergeben, die die Daten verschlüsselt überträgt.

Beispiel: Bei der Konfliktlösung stellt eine Partei eine Verbindung zum Vermittler her und schickt ihm die benötigten Daten. Dies kann zum Beispiel wie folgt geschehen:

```
TransactionID transaction_id = ...;
String ttp_url = "...";
Connection ttp_con = new ServerConnection(ttp_url, TTPServer.TTPPORT);
ttp_con.writeObject(transaction_id);
```

Über eine `ServerConnection` wird hier die Transaktionsnummer an den Vermittler geschickt. Das Java-Objekt `TransactionID` wird dabei serialisiert und vom Vermittler mittels `client_con.readObject()` empfangen.

4.2.3 Zeitbegrenzungen

Bei der Modellierung der in Kapitel 3 vorgeschlagenen Protokolle wird in Abschnitt 2.2.2 ein asynchrones Systemmodell vorausgesetzt, d.h. die Protokolle sind nicht abhängig von den Laufzeiten der Nachrichten oder den Rechengeschwindigkeiten der beteiligten Parteien. Daher wird für die Implementierung der Austauschprotokolle keinerlei Zeitinformation benötigt.

Aus praktischer Sicht gibt es jedoch mehrere Gründe, die für zeitliche Begrenzungen bei einem Fairneßdienst sprechen:

- Der Vermittler muß für jeden Austausch, an dem er beteiligt war, alle relevanten Daten für weitere Anfragen von einer der Parteien bereithalten [Aso98, S.34].

Um diesen Speicheraufwand zu reduzieren, sollten der Vermittler sehr alte Daten auslagern oder löschen können.

- Die Sicherheit der beim Austausch verwendeten kryptographischen Mechanismen ist nicht auf unbegrenzte Zeit garantiert (z.B. könnten die verwendeten Signaturschlüssel irgendwann unsicher werden). Daher sollte der Vermittler einen sehr alten Austausch nicht mehr fortsetzen.

Die praktische Implementierung eines Fairneßdienstes sollte also nur für einen begrenzten Zeitraum auf Anfragen bezüglich eines Austausches reagieren. Die Parteien haben dann nicht mehr beliebig lange Zeit, um Nachrichten an den Vermittler zu senden, sondern sie müssen immer die gesamte Kommunikation vor dem Ablauf des Zeitlimits beendet haben. Daher muß der Zeitraum hinreichend lange gewählt sein, so daß das Protokoll nicht durch das Zurückweisen von Nachrichten unfair wird.

Um den genauen Zeitraum für alle Parteien überprüfbar zu machen, sollte der Startzeitpunkt auch in die Transaktionsnummer

$$TID = H(ProtID, V, Startzeit, PubKey_A, PubKey_B, Rand_A, Rand_B, desc_{O_A}, desc_{O_B})$$

aufgenommen werden. Auf diese Weise kann dieser Zeitpunkt später nicht mehr manipuliert werden. Der Vermittler würde dann immer für eine bestimmte Zeitspanne (z.B. 6 Monate) für Anfragen bereitstehen. Nach Ablauf dieser Frist würde er Anfragen zu dieser Transaktion als veraltet zurückweisen.

Um jede Angriffsmöglichkeit auszuschließen, muß der Vermittler die Zeitspanne neu beginnen lassen, nachdem er eine Zustandsänderung bei einer Partei veranlaßt hat. Ansonsten besteht z.B. bei T0-Terminierung die Gefahr, daß die Partei A kurz vor Ablauf der Zeitspanne den Austausch mit dem Vermittler zu Ende führt und dadurch O_B bekommt, während B das Objekt O_A kaum innerhalb der verbleibenden Zeitspanne beim Vermittler abrufen kann. Durch die vorgeschlagene Maßnahme ist dagegen sichergestellt, daß jeder Partei stets die volle Zeitspanne für die Kommunikation mit dem Vermittler zur Verfügung steht.

Die hier beschriebene Lösung ist besonders effizient, da lediglich ein weiterer Wert bei der Hashwertberechnung zu berücksichtigen ist. Dagegen greift die von Asokan vorgeschlagene Lösung [Aso98, S.34] auf Zeitstempel zurück, was einen enormen Zusatzaufwand verursacht.

In der prototypischen Implementierung von FlexiFair gibt es vorerst keine Zeitbegrenzung, da die Probleme, die eine zeitliche Begrenzung erforderlich machen, bei einem Prototypen nicht schwerwiegend sind.

4.3 Beschreibung und Überprüfung von Objekten

In den vorhergehenden Abschnitten wurde immer vorausgesetzt, daß für jedes Objekt eine Beschreibung existiert, mit der sich überprüfen läßt, ob das gelieferte Objekt den

Erwartungen entspricht. In diesem Abschnitt werden verschiedene Konzepte für die Realisierung von Objektbeschreibungen vorgestellt und deren Vor- und Nachteile bei der Überprüfung von Objekteigenschaften diskutiert.

Bei der *Beschreibung* handelt es sich um alle Daten, die die Anforderungen an ein digitales Objekt spezifizieren. Das *Überprüfen*, ob ein gegebenes Objekt der Beschreibung entspricht, wird mittels einer *Prüffunktion* durchgeführt. Diese Prüffunktion benötigt als Eingabe wenigstens das zu prüfende Objekt und die Beschreibung, die die Anforderungen an das Objekt spezifiziert. Das Ergebnis eines Aufrufs der Prüffunktion ist `true`, falls das angegebene Objekt alle Anforderungen erfüllt, oder ansonsten `false`.

Die Implementierung der Beschreibung und Prüffunktion ist bei FlexiFair im Interface `Description` zusammengefaßt. Die Prüffunktion wird dabei durch die in `Description` definierte Methode `verifyItem` realisiert. Die Beschreibung entspricht den Daten, die in von `Description` abgeleiteten Klassen enthalten sind. Eine solche abgeleitete Klasse ist zum Beispiel `SignatureDescription`, die die Signatur einer Nachricht mit einem bestimmten Schlüssel spezifiziert.

Beispiel: Im folgenden Beispiel wird überprüft, ob ein empfangenes Objekt der Beschreibung durch die Klasse `SignatureDescription` entspricht.

```
Description desc = new SignatureDescription(...);
ExchangeableItem signature = (ExchangeableItem) ttp_con.readObject();
if (desc.verifyItem(signature, transaction_id))
   ...; // Empfangenes Objekt ist gültige Signatur
else
   ...; // Objekt entspricht nicht der Beschreibung
```

Das Objekt `signature` der Klasse `ExchangeableItem` enthält das empfangene Objekt und zusammen mit der Transaktionsnummer dienen beide als Eingabe zum Überprüfen, ob eine der Beschreibung `SignatureDescription` entsprechende Signatur empfangen wurde.

Beim Einsatz solcher von `Description` abgeleiteten Klassen muß zwischen einem statischen Ansatz mit vordefinierten Klassen (wie z.B. `SignatureDescription`) und einem flexiblen Ansatz mit dynamisch zur Laufzeit bereitgestellten Beschreibungen unterschieden werden.

4.3.1 Statischer Ansatz

In der Implementierung von FlexiFair gibt es vordefinierte Klassen, die die Beschreibung und Überprüfung von bestimmten Objekten ermöglichen. Beispiele für solche vordefinierten Klassen sind Beschreibungen für Signaturen (`SignatureDescription`), Verträge (`GenContractDescription`) oder Objekte mit einem Nachweis der Herkunft

(`NRODescription`). Diese Klassen sind beim Vermittler und den am Austausch beteiligten Parteien vorhanden und werden aufgerufen, sobald eine Beschreibung erzeugt und eine Überprüfung durchgeführt werden soll. Die korrekte Funktionsweise dieser Klassen wird durch eine sorgfältige Implementierung und eine gründliche Begutachtung dieser Klassen sichergestellt.

Vorteile: Die vordefinierten Klassen werden von allen Anwendungen eingesetzt, die FlexiFair nutzen. Dies erleichtert und unterstützt die Weiterentwicklung dieser Klassen, da die Funktionsfähigkeit und Stabilität der Klassen beim Einsatz in vielen Anwendungen getestet wird.

Nachteile: Die vorgegebenen Klassen schränken die Anwendungsmöglichkeiten von FlexiFair stark ein. Wenn eine Anwendung den Fairneßdienst FlexiFair nutzen möchten und keine geeignete Klasse zur Beschreibung und Überprüfung der von dieser Anwendung ausgetauschten Objekte vorfindet, erweist sich der statische Ansatz als zu unflexibel: Der Programmierer der Anwendung muß seine eigene Klassen für die Beschreibung der ausgetauschten Objekte schreiben. Schließlich muß der Vermittler dazu gebracht werden, diese neuen Klassen zu installieren und somit seinen Dienst zu erweitern. Da ein Vermittler jedoch die Korrektheit der Protokollausführung sicherstellen muß, ist das Installieren von fremden Klassen ein Sicherheitsrisiko und muß daher sorgfältig geprüft werden, was einen großen Aufwand an Zeit und Geld verursacht.

4.3.2 Dynamischer Ansatz

Um die Hürden für den Einsatz von FlexiFair in neuen Anwendungen deutlich zu verringern, habe ich den flexibleren dynamischen Ansatz zur Erweiterung von FlexiFair entwickelt (siehe auch [PVGW00] unter dem Begriff „Sandboxing“). Dieser Ansatz besteht aus der in den folgenden Schritten beschriebenen Vorgehensweise:

1. Ein Anwendungsprogrammierer implementiert die für seine Anwendung benötigten Klassen zur Beschreibung und Überprüfung von Objekten.
2. Die Anwendung verwendet diese Klassen zur Durchführung des fairen Austausches.
3. Sobald der Vermittler am Austausch beteiligt wird, erhält er die ihm bisher unbekannten Klassen übermittelt.
4. Die neuen Klassen werden vom Vermittler in einer *geschützten Ausführungsumgebung* (engl. Sandbox) ausgeführt. Diese beschränkt die Ausführungsrechte der Klassen, so daß nur unbedenkliche Befehle ausgeführt werden können.

Mit diesem dynamischen Ansatz können neue Anwendungen auf die vom Vermittler bereitgestellten Dienste zugreifen, ohne daß für jede neue Anwendung neue Klassen beim Vermittler installiert werden müssen. Stattdessen werden die neuen Klassen dynamisch vom Vermittler geladen und ausgeführt.

4.3.3 Sicherheitsanforderungen

Bei der Beschreibung und Überprüfung von Objekten durch eine Klasse, die `Description` implementiert, müssen unabhängig vom verwendeten Implementierungsansatz die folgenden Anforderungen erfüllt sein, damit ein Austausch fair bleibt:

Konsistenz: Die Methode `verifyItem` muß bei allen Aufrufen mit gleicher Eingabe immer das gleiche Ergebnis liefern.

Kontrollierter Informationsfluß: Die Methode `verifyItem` darf – außer dem Rückgabewert `true`/`false` – keine Informationen über das geprüfte Objekt an irgendjemanden übermitteln.

Durch die erste Anforderung soll sichergestellt werden, daß bei gleicher Beschreibung und gleichem Objekt die Überprüfung das gleiche Ergebnis liefert. Insbesondere soll dadurch ausgeschlossen werden, daß eine Überprüfung zufällige Ergebnisse liefert, daß das Ergebnis zeitabhängig ist oder daß das Ergebnis vom Rechner abhängt, auf dem die Überprüfung durchgeführt wird. Ein solches Verhalten würde sonst unvorhersehbare Auswirkungen auf die Durchführung des fairen Austausches haben.

Die zweite Anforderung ist notwendig, damit nicht unberechtigterweise Daten über die auszutauschenden Objekte weitergegeben werden. Falls ein Austausch abgebrochen wird, garantiert diese Anforderung, daß A und B keine neuen Informationen über die gewünschten Objekte erhalten haben.

Bei den in FlexiFair vordefinierten Klassen sind diese Anforderungen natürlich immer erfüllt. Dagegen gibt es bei der dynamischen Erweiterung um neue Klassen das Problem, daß der Programmierer der neuen Klassen diese Regeln nicht einhalten muß. Daher müssen gerade beim dynamischen Ansatz diese Sicherheitsanforderungen kontrolliert und durchgesetzt werden.

4.3.4 Sicherheit bei der Implementierung von Objektbeschreibungen

Statischer Ansatz

Bei der Implementierung des statischen Ansatzes können die beiden Sicherheitsanforderungen leicht erfüllt werden. Die vordefinierten Klassen zur Beschreibung und Überprüfung von Objekten sind öffentlich bekannt und können von jedem daraufhin geprüft werden, ob diese Klassen die Sicherheitsanforderungen einhalten. Sowohl für den Vermittler, der sich verpflichtet hat, nur diese öffentlich bekannten Klassen zu verwenden, als auch für die jeweilige Partei selbst, die ebenfalls diese Klassen verwendet, ist damit sichergestellt, daß keine Verletzung der Konsistenz und des kontrollierten Informationsflusses auftritt. Die jeweils andere am Austausch beteiligte Partei kann zwar eine eigene Implementierung verwenden, was jedoch aufgrund der Fairneß des verwendeten Protokolls höchstens zu einer Benachteiligung dieser Partei führen kann.

Dynamischer Ansatz

Beim dynamischen Ansatz mit seinen proprietären, nicht öffentlich bekannten Klassen muß ein deutlich größerer Aufwand getrieben werden, um die Sicherheitsanforderungen einzuhalten.

Die Forderung nach Konsistenz ist insbesondere bei optimistischen Protokollen kritisch: Die Überprüfung eines Objekts könnte zum Beispiel beim Vermittler immer scheitern, während die Überprüfung des gleichen Objekts durch eine andere Partei erfolgreich ist. Dann würde sich der Vermittler bei generierbaren Objekten immer weigern, den Austausch fortzusetzen, so daß eine Partei keine Konfliktlösung durchführen kann, was zu einer klaren Benachteiligung führt.

Dieses Problem durch inkonsistente Antworten bei der Überprüfung von Objekten kann dagegen bei aktiven Protokollen nicht auftreten. Da der Vermittler atomar die Entscheidung über die Fortführung oder den Abbruch des Austausches trifft, ist eine konsistente Antwort automatisch gewährleistet. Bei aktiven Protokollen treten also keine Überraschungen auf, weil solche Protokolle nur eine einzige Überprüfung durchführen.

Die zweite Bedingung, die keinen Informationsfluß fordert, kann nur realisiert werden, wenn die verwendete Programmiersprache eine geschützte Ausführungsumgebung für fremde Programme zur Verfügung stellt. Die Programmiersprache Java bietet eine solche *geschützte Ausführungsumgebung* für fremde Programme an (z.B. beim Laden von Applets über das Internet) und kann so den Informationsfluß verhindern. Bei der Implementierung muß lediglich darauf geachtet werden, daß die fremden Klassen über einen anderen Mechanismus als die lokalen Klassen geladen werden. Diese Aufgabe übernimmt bei FlexiFair die speziell dafür entwickelte Klasse `JarClassLoader`, die eine beliebige Anzahl fremder Klassen aus einer mitgelieferten Jar-Datei heraus instanzieren kann. Alle fremden Klassen, die über `JarClassLoader` instanziert werden, haben automatisch keine Rechte für den Zugriff auf Netzwerkverbindungen oder das Dateisystem. Damit ist auch sichergestellt, daß bösartige Programme keinen Schaden anrichten können.

4.4 Digitale Objekte

Wie bereits in Abschnitt 2.3.4 beschrieben, bestehen digitalen Objekte im Prinzip aus einem einzelnen Bitstring. Daher lassen sie sich mit einer einzigen Nachricht verschicken und bei Bedarf auch vollständig zwischenspeichern. Diese Eigenschaft entspricht in der Programmiersprache Java der Serialisierbarkeit von Objekten, so daß digitale Objekte durch eine beliebige serialisierbare Klasse implementiert werden können.

Bei der Implementierung im Rahmen von FlexiFair dient die serialisierbare Klasse `ExchangeableItem` als Container für das digitale Objekt, welches mit den Methoden `getValue` und `setValue` gelesen und gesetzt werden kann.

Beispiel: Im diesem Beispiel ist `ExchangeableItem` der Container für eine Signatur, die in Java aus einem Byte-Array besteht. Diese Signatur-Bytes werden ausgelesen und überprüft.

```
ExchangeableItem signature = (ExchangeableItem) ttp_con.readObject();
byte[] sig_bytes = (byte[]) signature.getValue();
Signature s = Signature.getInstance(...);
s.initVerify(...); s.update(...);
if (s.verify(sig_bytes))
   ...; // Es handelt sich um eine gültige Signatur
else
   ...; // Keine gültige Signatur
```

Die Objekteigenschaften der Generierbarkeit und Widerrufbarkeit stellen eine Erweiterung der Klasse `ExchangeableItem` dar, die in den folgenden beiden Abschnitten detailiert beschrieben wird. Auch die Nachweise des Empfangs und der Herkunft, die bei Protokollen mit Nichtabstreitbarkeit des Empfangs und der Herkunft zusätzlich ausgetauscht werden, werden in den Abschnitten 4.4.3 und 4.4.4 als eine Erweiterung der Objekte modelliert.

4.4.1 Generierbarkeit

Eine Erweiterung beliebiger digitaler Objekte sind generierbare Objekte, die durch die Klasse `GeneratableItem` repräsentiert werden. Diese Klasse stellt zusätzlich die Methode `generateItemByTTP` bereit, mit der der Vermittler das Objekt generieren kann. Beim Generieren müssen die folgenden Voraussetzungen erfüllt sein:

- Die Partei, die das Objekt generiert haben möchte, muß dem Vermittler die benötigten Generierungsinformationen liefern.
- Der Vermittler muß geheime Zusatzinformationen besitzen, die nur ihm das Generieren des Objekts ermöglichen.
- Das zu generierende Objekt muß nachweislich zur aktuellen Austauschtransaktion gehören.

Die *Generierungsinformation* wird von der Partei, die das generierbare Objekt austauschen will, bereitgestellt und wird dann von einem Protokoll, das mit generierbaren Objekten arbeitet, während der Durchführung von Modul M2 übertragen. Der Empfänger muß sich dann von der Korrektheit dieser Generierungsinformation überzeugen können. Daher gibt es eine Erweiterung der Beschreibung und Überprüfung speziell für generierbare Objekte. Das Interface `GeneratableItemDescription` stellt die zusätzliche Methode `verifyGenInfo` bereit, mit der der Empfänger der Generierungsinformation deren Gültigkeit überprüfen kann. Bei starker Generierbarkeit garantiert eine erfolgreiche Überprüfung, daß der Vermittler mit dieser Information das gewünschte Objekt generieren kann. Dagegen stellt eine erfolgreiche Überprüfung der Generierungsinformation bei

schwacher Generierbarkeit lediglich sicher, daß der Vermittler, falls er beim Generieren scheitert, immer den Erzeuger dieser Generierungsinformation als Betrüger identifizieren wird.

Damit die Generierungsinformation nicht in anderen Austauschtransaktionen verwendet werden kann, muß sie, wie in Abschnitt 4.1.2 beschrieben, an die aktuelle Transaktionsnummer gebunden sein. Dies wird dann ebenfalls von der Methode `verifyGenInfo` geprüft.

Wenn der Vermittler aus einer Generierungsinformation ein Objekt generieren möchte, benötigt er als Zusatzinformation noch ein Geheimnis vom Typ `GenerateSecret`. Welches Geheimnis genau benötigt wird, verrät die Methode `getGenSecretName` der Beschreibung `GeneratableItemDescription`. Der Zugriff auf eine Instanz der benötigten geheimen Zusatzinformation erhält der Vermittler über ein Objekt vom Typ `TTPSecrets`, das mit der Methode `getSecret` die gewünschten Geheimnisse bereitstellt. Die Methode `generateItemByTTP` der Klasse `GeneratableItem` verwendet schließlich diese geheime Information, um aus der Generierungsinformation das Objekt zu generieren.

Beispiele

Starke Generierbarkeit. Bei elektronischen Verträgen kann starke Generierbarkeit zum Beispiel besonders einfach realisiert werden (siehe auch Abschnitt 4.7.1 für das vollständige Szenario des Vertragsaustausches). Als Generierungsinformation wird eine vom gewünschten Vertragspartner unterschriebene Absichtserklärung benötigt, in der dieser seinen Willen ausdrückt, den Vertrag zu unterschreiben, falls er im Gegenzug den von der anderen Partei signierten Vertrag erhält. Die Generierungsinformation ist also eine Signatur der folgenden Daten:

- Transaktionsnummer: Damit wird die Verknüpfung zur aktuellen Austauschtransaktion hergestellt.
- Generierbarkeitserklärung: Es wird festgelegt, daß die Partei dem Vermittler erlaubt, den entsprechenden Vertrag bei Bedarf zu generieren.
- Geplanter Vertragstext: Die eigentliche Aussage des auszutauschenden Vertrags.

Der Vermittler kann dann aus diesen Daten einen Ersatzvertrag konstruieren, falls die für den Vertragsaustausch genutzte Anwendung diesen akzeptiert. Dieser Ersatzvertrag besteht einfach aus einer Unterschrift des Vermittlers auf die Signatur der Absichtserklärung. Dieser Vertrag ist genau dann gültig, wenn die Signaturen des Vermittlers und der Vertragspartei gültig sind.

In diesem Beispiel hat der generierte Vertrag eine andere Form als der direkt von den Parteien ausgetauschte Vertrag. Bei solchen Objekten kann eine Anwendung die Mitwirkung des Vermittlers erkennen, indem sie nach der Überprüfung des generierbaren Objekts durch `verifyItem` die Methode `generatedByTTP` von `GeneratableItemDescription` aufruft. Auf diese Weise lassen sich die Aktionen des Vermittlers teilweise überprüfen. Wenn zum Beispiel der Vermittler fehlerhaft arbeiten

würde und sowohl ein Objekt generiert als auch bei einer anderen Anfrage zur gleichen Transaktion den Austausch als abgebrochen bezeichnet, dann könnte man diese widersprüchlichen Antworten des Vermittlers entdecken und dessen Fehlverhalten nachweisen [ASW98a].

Schwache Generierbarkeit. Jedes digitale Objekt kann wie folgt auf einfache Weise schwach generierbar gemacht werden. Die Generierungsinformation wird von der Partei erzeugt, die das Objekt bereitstellen will und besteht aus einer Signatur über die folgenden Informationen:

- Transaktionsnummer: Damit wird die Verknüpfung zur aktuellen Austauschtransaktion hergestellt.
- Generierbarkeitserklärung: Die erstellende Partei verpflichtet sich, die folgenden Daten so zu berechnen, daß der Vermittler daraus das Objekt generieren kann.
- Verschlüsselung der folgenden Werte mit dem öffentlichen Schlüssel des Vermittlers:
  - Transaktionsnummer und
  - das auszutauschende Objekt

Die schwache Generierbarkeit wird überprüft, indem man die Signatur verifiziert. Welche Werte verschlüsselt wurden kann dabei nicht überprüft werden, was jedoch bei schwacher Generierbarkeit auch nicht nötig ist.

Der Vermittler generiert das Objekt, indem er es entschlüsselt. Falls kein gültiges Objekt verschlüsselt war oder die Transaktionsnummern nicht mit der aktuellen Austauschtransaktion übereinstimmen, wird der Vermittler den Generierungsversuch als gescheitert abbrechen. Aufgrund der Signatur weiß der Vermittler, daß der Erzeuger dieser Generierungsinformation nicht korrekt gearbeitet hat.

4.4.2 Widerrufbarkeit

Widerrufbare Objekte sind eine Erweiterung von beliebigen digitalen Objekten und werden von der Klasse `RevocableItem` implementiert. Diese Klasse erlaubt dem Vermittler, mit der Methode `revokeItemByTTP` dieses Objekt zu widerrufen. Auf diese Weise führt der Vermittler eine Aktion aus, die das widerrufbare Objekt für den beim Austausch vorgesehenen Empfänger wertlos macht. Dafür müssen die folgenden Anforderungen erfüllt sein:

- Der Vermittler muß das widerrufbare Objekt erhalten und dessen Gültigkeit überprüft haben.
- Das widerrufbare Objekt muß nachweislich zur aktuellen Austauschtransaktion gehören.

- Der Vermittler muß geheime Zusatzinformation besitzen, die nur ihm das Widerrufen des Objekts ermöglichen.

Das Empfangen und Prüfen eines widerrufbaren Objekts durch den Vermittler wird vom Austauschprotokoll durchgeführt, sofern dies auf Widerrufbarkeit angewiesen ist. Der Vermittler darf jedoch kein Objekt widerrufen, das in einer anderen Austauschtransaktion verwendet wird. Daher muß bei einem widerrufbaren Objekt immer auch geprüft werden, ob es zur aktuellen Austauschtransaktion gehört. In dem Objekt muß die Transaktionsnummer so integriert sein, daß eine erneute Verwendung in einer anderen Austauschtransaktion ausgeschlossen ist. Eine solche Funktionalität kann mit den in Abschnitt 4.1.2 beschriebenen Verfahren bereitgestellt werden. Die Überprüfung der Transaktionsnummer geschieht dann während der Überprüfung des Objekts mittels der entsprechenden Beschreibung für das widerrufbare Objekt.

Wenn der Vermittler ein Objekt widerrufen möchte, benötigt er als Zusatzinformation noch ein Geheimnis vom Typ `RevokeSecret`. Welches Geheimnis genau benötigt wird, verrät die Methode `getRevSecretName` des widerrufbaren Objekts. Der Zugriff auf eine Instanz der benötigten geheimen Zusatzinformation erhält der Vermittler über ein Objekt vom Typ `TTPSecrets`, das mit der Methode `getSecret` die gewünschten Geheimnisse bereitstellt. Die Methode `revokeItemByTTP` der Klasse `RevocableItem` verwendet schließlich diese geheime Information, um das Objekt zu widerrufen.

Beispiele

Starke Widerrufbarkeit. Elektronische Bezahlverfahren können beispielsweise starke Widerrufbarkeit zur Verfügung stellen. Bei einem Scheck-ähnlichen Zahlungssystem kann die Zahlung der Partei A an B aus einer Signatur $sig_A(TID, \text{Zahlung von } A \text{ an } B, Betrag)$ bestehen. Die Bank bucht beim Einreichen einer solchen Signatur den genannten Betrag vom Konto der Partei A ab und schreibt das Geld der Partei B gut.

Falls der Vermittler beim Austausch mit der Transaktionsnummer TID eine solche Zahlung widerrufen muß, erstellt er eine Signatur $sig_V(TID, \text{Widerruf}, A)$ und leitet diese an die Bank von A weiter. Diese wird dann die zu dieser Transaktionsnummer gehörende Buchung rückgängig machen. Falls noch keine Buchung vorgenommen wurde, wird diese Transaktionsnummer von der Bank nicht mehr für Zahlungen von A akzeptiert.

Schwache Widerrufbarkeit. Ein Flugticket, das von einem Reisebüro verkauft wird, ist ein Beispiel für schwache Widerrufbarkeit. Wenn der Vermittler ein solches Ticket widerrufen möchte, informiert er die Fluggesellschaft, die bis zum Abflug das Ticket noch für ungültig erklären kann. Falls das Flugticket bereits verwendet wurde, scheitert der Widerruf und die Fluggesellschaft teilt dies dem Vermittler mit.

Ein bei einem Reisebüro R gebuchtes Ticket könnte dann aus einer Signatur

$$sig_R(TID, \text{Reisebuchung}, sig_F(TID, \text{Flugticket}, Flugdaten))$$

bestehen, die auch eine Bestätigung der Buchung durch die Fluggesellschaft F enthält. Der Vermittler könnte dieses Ticket widerrufen, indem er eine Signatur $sig_V(TID,$

Widerruf, $Flugdaten$) an die Fluggesellschaft schickt. Falls diese das Flugticket widerrufen kann, sendet sie eine Signatur $sig_F(TID, \text{Widerrufen}, Flugdaten)$. Wenn das Widerrufen scheitert, könnte die Antwort aus der Signatur $sig_F(TID, \text{nicht widerrufbar}, Flugdaten)$ bestehen.

4.4.3 Nichtabstreitbarkeit der Herkunft

In der Literatur wird normalerweise die Nichtabstreitbarkeit der Herkunft als eine zusätzliche Eigenschaft des Austauschprotokolls angesehen. Aus der Implementierungssicht stellt sich diese Eigenschaft jedoch anders dar: Die Nichtabstreitbarkeit der Herkunft sollte besser als eine Objekteigenschaft modellieren werden. Anstatt lediglich ein Objekt O_P auszutauschen, wird einfach das Objekt mit dazugehörigem Nachweis der Herkunft verwendet, also O_P, $\text{NH}(P, O_P)$. Daher muß die Partei P, die das Objekt O_P bereitstellt, gleichzeitig auch den Nachweis der Herkunft erzeugen und mitliefern. Ein derart erweitertes Objekt wird genau dann akzeptiert, wenn es die Beschreibung erfüllt und einen gültigen Nachweis der Herkunft enthält. Der Einfachheit halber fasse ich diese beiden Anforderungen in einer erweiterten Beschreibung zusammen, die speziell für Objekte mit Nachweis der Herkunft verwendet wird.

Der Vorteil der vorgeschlagenen Erweiterung liegt auf der Hand: *Jedes* Austauschprotokoll kann wahlweise Objekte mit und ohne Nachweis der Herkunft austauschen. Es kann sogar für ein Objekt O_P dieser Nachweis gefordert werden, während beim anderen Objekt O_Q darauf verzichtet wird. Dies ermöglicht maximale Flexibilität beim Einsatz von Austauschprotokollen, was die folgenden Beispiele verdeutlichen:

- Kein Nachweis der Herkunft: Wenn zum Beispiel digital signierte Verträge ausgetauscht werden, sind die beteiligten Personen immer bekannt und nachprüfbar. Zusätzliche Nachweise der Herkunft hätten also keinen Nutzen und würden nur die Effizienz durch die überflüssigen Überprüfungen verringern.

- Einseitiger Nachweis der Herkunft: Bei einem anonymen Einkauf ist ein Nachweis der Herkunft nur für die Ware sinnvoll. Die Zahlung sollte keinen solchen Nachweis enthalten, weil sie anonym erfolgen soll. Falls ein Pseudonym verwendet wird, würde ein unter diesem Pseudonym erstellter Nachweis der Herkunft die Anonymität der Zahlung im Prinzip nicht verletzen. Dann besitzt der Nachweis aber praktisch keine Aussagekraft mehr, was ihn überflüssig für die Nichtabstreitbarkeit der Herkunft macht.

- Beidseitiger Nachweis der Herkunft: Ein anderes Beispiel ist eine Empfangsbestätigung für eine E-Mail. Diesmal empfiehlt es sich, sowohl für den Empfänger der E-Mail als auch für den Empfänger der Empfangsbestätigung, einen Nachweis der Herkunft zu verlangen.

Mit der beschriebenen Modellierung können diese verschiedenen Szenarien alle das gleiche Austauschprotokoll verwenden. Durch die Verwendung von erweiterten Beschreibungen werden Nachweise der Herkunft automatisch von den Protokollen berücksichtigt.

Implementierung

Bei der Implementierung von Nichtabstreitbarkeit der Herkunft im Rahmen von FlexiFair werden Erweiterungen für Objekte und Beschreibungen bereitgestellt. Grundsätzlich werden Nachweise für die Herkunft von Objekten durch Signaturen der Form $Sig_P(TID, \mathrm{NH}, O_P)$ implementiert, wobei NH angibt, daß diese Signatur als ein Nachweis der Herkunft für das Objekt O_P interpretiert werden muß. In der entsprechenden Beschreibung `NRODescription` (NRO steht für non-repudiation of origin) ist festgelegt, welcher Schlüssel und Signaturalgorithmus für diesen Nachweis der Herkunft verwendet werden soll.

Die Objekte können als neue Eigenschaft die Nichtabstreitbarkeit der Herkunft besitzen, welche durch das Interface `NRO` bereitgestellt wird. Dieses Interface stellt die Methoden `setNRO` und `getNRO` zur Verfügung. In `setNRO` wird eine Signatur als Nachweis der Herkunft gespeichert. Mit `getNRO` kann dann auf diese Signatur zugegriffen werden.

Wenn bei einem Austauschvorgang auch Nachweise der Herkunft ausgetauscht werden sollen, müssen die folgenden Dinge beachtet werden:

- Die Objekte, die mit einem Nachweis der Herkunft ausgetauscht werden, müssen das Interface `NRO` implementieren. Der Lieferant dieses Objekts erzeugt dann einen Nachweis der Herkunft, der mit der Methode `setNRO` in dem Objekt gespeichert und dann zusammen mit dem Objekt verschickt wird.
- Die Beschreibungen für ein Objekt mit Nachweis der Herkunft müssen in der Methode `verifyItem` auch die Gültigkeit des mit dem Objekt gelieferten Nachweises prüfen. Diese Beschreibung legt auch fest, mit welchem Algorithmus und Schlüssel die Signatur für den Nachweis der Herkunft erstellt sein muß.

4.4.4 Nichtabstreitbarkeit des Empfangs

Zur Realisierung eines Nachweises des Empfangs werden zuerst einmal Austauschprotokolle benötigt, die einen solchen Nachweis erzeugen. Die Konzepte für solche Protokolle wurden in Abschnitt 3.2 beschrieben.

Im Gegensatz zur Nichtabstreitbarkeit der Herkunft ist die Bedeutung der Nichtabstreitbarkeit des Empfangs auf die aktuell durchgeführte Austauschtransaktion begrenzt. Der Nachweis des Empfangs ist also nur für genau diesen Austausch gültig und kann nicht im Zusammenhang mit einem anderen Austausch verwendet werden. Aus diesem inhaltlichen Zusammenhang ergeben sich eine Reihe von Anforderungen:

- Der Nachweis des Empfangs muß über die eindeutige Transaktionsnummer an den Austausch gekoppelt sein.
- Der Nachweis sollte nur im Zusammenhang mit diesem Austausch eine Bedeutung haben.
- Der Nachweis muß bei Bedarf vom Vermittler erzeugt werden können.

Bei der Implementierung von Protokollen mit Nachweis des Empfangs werden jeweils Signaturen der Form $Sig_P(TID, \text{NE}, O_Q)$ über die empfangenen Objekte erzeugt und an die andere Partei geschickt. Aufgrund der zweiten Anforderung wird dazu immer der für diesen Austausch vereinbarte Signaturschlüssel verwendet. Der Nachweis besteht dann aus einer Signatur über das empfangene Objekt und die Transaktionsnummer, so daß die Koppelung an diesen Austausch gewährleistet ist.

Der Vermittler kann während einer Konfliktlösung diese Daten auch mit seinem Signaturschlüssel signieren und so einen Ersatz $Sig_V(TID, \text{NE-P}, O_Q)$ für einen fehlenden Nachweis des Empfangs von P schaffen.

Implementierung

Ein Nachweis des Empfangs wird durch die Klasse `NRRToken` repräsentiert (NRR steht für non-repudiation of receipt). Diese enthält die Methoden `setToken` und `getToken`, mit denen die Signatur gespeichert und gelesen werden kann.

Obwohl die Nichtabstreitbarkeit des Empfangs durch die verwendeten Protokolle gewährleistet werden muß, werden auch die Objekte erweitert, um einen Nachweis für den Empfang beinhalten zu können. Die Objekte können daher als neue Eigenschaft die Nichtabstreitbarkeit des Empfangs besitzen, welche durch das Interface `NRR` bereitgestellt wird. Dieses Interface stellt die Methoden `setNRR` und `getNRR` zur Verfügung, mit denen der Nachweis des Empfangs gespeichert und gelesen werden kann.

Da bei einem solchen Austausch mit Nachweis des Empfangs jede Partei das gewünschte Objekt zusammen mit dem Nachweis des Empfangs für das gesendete Objekt erhält, bezieht sich der in einem Objekt gespeicherte Nachweis des Empfangs auf das in diesem Austauschvorgang gesendete Objekt. Wenn ein Nachweis des Empfangs also überprüft werden soll, muß das gesendete Objekt vorhanden sein.

4.5 Austauschprotokolle

Die Implementierung der Austauschprotokolle folgt der Modellierung aus Abschnitt 2.3.3. Das Aushandeln der Protokollparameter ist dabei Aufgabe der Anwendung selbst, so daß nur die Module M2 bis M5 implementiert werden müssen. Die dafür benötigten Methoden definiert die abstrakte Klasse `ExchangeProtocol`. Die meisten Methoden sind in zwei Varianten vorhanden, da die austauschenden Parteien andere Methoden benötigen als der Vermittler. Dies beginnt bei der Initialisierung des Java-Objekts, über die Vorbereitung (Modul M2) und die Durchführung des Austausches (Modul M3) und geht bis zur Konfliktlösung (Modul M4/M5). Das folgende Beispiel zeigt den Einsatz des Austauschprotokolls `ActiveProtocol` aus der Sicht von einer der Parteien. Das Protokoll `ActiveProtocol` ist die Implementierung von P1 aus Abschnitt 3.1.2.

Beispiel: Zuerst wird das Protokoll ausgewählt und initialisiert. Dabei werden die Transaktionsnummer, der private Schlüssel zum Signieren von Nachrichten und ein Objekt vom Typ `Storage` übergeben. Der Parameter `true` gibt an, daß diese Partei die Rolle von A in dem Protokoll übernehmen wird. Dann wird der Austausch mit

der Methode `prepareExchangeClient` vorbereitet, wofür das eigene Objekt und Kommunikationsverbindungen vom Typ `Connection` benötigt werden. Falls dieser Schritt fehlschlägt, sollte der Austausch abgebrochen werden. Ansonsten wird der Austausch mit der Methode `doExchangeClient` durchgeführt, was das gewünschte Objekt liefern sollte. Wenn das nicht der Fall ist, muß die Konfliktlösung durch Fortsetzen des Austausches gestartet werden.

```
ExchangeProtocol ex_prot =
                        ExchangeProtocol.getInstance(‘‘ActiveProtocol’’);
ex_prot.initClient(transaction_id, priv_sig_key, true, storage);
if (ex_prot.prepareExchangeClient(own_item, client_con, ttp_con)) {
   received_item = ex_prot.doExchangeClient(client_con, ttp_con);
   if (received_item != null)
      ... // Austausch erfolgreich beendet
   else
      ... // Fehler! Konfliktlösung durch Fortsetzen (M4)
}
else
   ... // Fehler! Konfliktlösung durch Abbrechen (M5)
```

Wenn das Vorbereiten des Austausches fehlschlägt und der Austausch abgebrochen werden soll, kann wie folgt die Konfliktlösung gemäß Modul M5 gestartet werden:

```
if (ex_prot.abortExchangeClient(client_con, ttp_con))
   ... // Austausch wurde abgebrochen
else
   ... // Fehler! Konfliktlösung durch Fortsetzen (M4)
```

Das Fortsetzen des Austausches durch das Konfliktlösungsmodul M4 kann schließlich wie folgt durchgeführt werden:

```
received_item = ex_prot.resolveExchangeClient(client_con, ttp_con);
if (received_item != null)
   ... // Austausch erfolgreich beendet
else
   ... // Fehler! Konfliktlösung durch Abbrechen (M5)
```

An diesem Beispiel zeigt sich auch die Flexibilität dieser Modellierung von Austauschprotokollen. Wenn die ausgetauschten Objekte die Eigenschaften Generierbarkeit und/oder Widerrufbarkeit besitzen, können statt P1 auch einfach die optimistischen Protokolle P2 bis P5 eingesetzt werden. Zum Beispiel kann bei zwei generierbaren Objekten das Protokoll P2 verwendet werden, indem die Zeichenkette ‘‘`ActiveProtocol`’’ durch ‘‘`ASW98Protocol`’’ ersetzt wird. Weitere Änderungen sind nicht nötig!

4.6 Der Vermittler

Einen weiteren Teil von FlexiFair bildet der Vermittler, auf den über das Internet zugegriffen werden kann, um für eine faire Abwicklung des Austausches zu sorgen. Der Vermittler ist ein Server, der ebenfalls in Java programmiert ist. Wenn sich eine Partei an den Vermittler wendet, teilt sie ihm zuerst alle Informationen mit, die zur Berechnung der Transaktionsnummer benötigt werden. Dadurch kann der Vermittler eine alte Sitzung fortsetzen, die unter dieser Transaktionsnummer gespeichert ist. Falls er keine alte Sitzung findet, kennt er das gewünschte Fairneßprotokoll aus den für die Berechnung der Transaktionsnummer gesendeten Daten. Die entsprechende Implementierung dieses Protokolls übernimmt die weitere Durchführung des fairen Austausches.

Das Ziel des Vermittlers ist es, jede beliebige Anwendung beim fairen Austausch zu unterstützen. Dazu müssen die folgenden Softwarekomponenten vorab beim Vermittler installiert sein: Das Austauschprotokoll, die ausgetauschten Objekte und die Beschreibungen der Objekte. Durch die Bereitstellung eines aktiven Protokolls wird sichergestellt, daß jede Anwendung zumindest dieses Austauschprotokoll problemlos nutzen kann, da aktive Protokolle keine besonderen Anforderungen an die ausgetauschten Objekte stellen. Entsprechend können dann auch einfache Objekte vom Typ `ExchangeableItem` verwendet werden, weil diese als Container für beliebige Daten dienen können. Durch den in Abschnitt 4.3.2 beschriebenen dynamischen Ansatz, mit dem Beschreibungen bei aktiven Protokollen zur Ausführungszeit an den Vermittler übergeben werden können, kann schließlich jede Anwendung eigene Beschreibungen verwenden, falls die beim Vermittler vorinstallierten nicht ausreichen. Auf diese Weise kann also jede Anwendung den vom Vermittler bereitgestellten Austauschdienst nutzen.

Für häufig genutzte Anwendungen ist es allerdings von Vorteil, wenn auf optimistische Protokolle zurückgegriffen werden kann. Dann wird der Vermittler bei einer fehlerfreien Ausführung des Austausches nicht benötigt, wodurch dessen Last deutlich vermindert werden kann. Daher sollte bei der Entwicklung von Anwendungen, die fairen Austausch nutzen möchten, immer darauf geachtet werden, daß möglichst ein Objekt starke Generierbarkeit oder Widerrufbarkeit besitzt.

4.7 Beispiele für den Einsatz des Fairneßdienstes

In diesem Abschnitt zeige ich, wie sich Problemstellungen aus verschiedenen Anwendungsbereichen mit Hilfe von FlexiFair einfach und flexibel lösen lassen. Eine Anwendung muß nicht selbst eine Lösung für den fairen Austausch implementieren, sondern greift auf

die generisch implementierten Protokolle von FlexiFair zurück. Dadurch reduziert sich die ursprüngliche Problemstellung auf die Auswahl der geeigneten Protokolle und die Abbildung der ausgetauschten Objekte auf die von FlexiFair bereitgestellte Infrastruktur. Wie solche Abbildungen konkret aussehen können, wird in den folgenden Beispielen ausführlich beschrieben.

4.7.1 Vertragsunterzeichnung

Bei einer elektronischen Vertragsunterzeichnung vereinbaren die Parteien A und B, daß sie beide einen bestimmten Vertragstext digital signieren und dann diese Unterschriften austauschen wollen. Beide Parteien wollen sicherstellen, daß die andere Partei ihre Unterschrift nur genau dann erhält, wenn sie im Gegenzug die Unterschrift der anderen Partei erhalten. Andernfalls müßte eine Partei fürchten, daß sie durch ihre Unterschrift an den Vertrag gebunden ist, ohne zu wissen, ob die andere Partei diesen Vertrag unterzeichnet hat.

Beim Aushandeln in Modul M1 muß zusätzlich zu den üblichen Daten noch die folgende Informationen bekanntgegeben werden:

Öffentliche Information: Der Vertragstext.

Die auszutauschenden Unterschriften können nun wie folgt als stark generierbare Objekte modelliert werden. Da beide Parteien die gleiche Art von Objekten verwenden, gelten diese Ausführungen für jede Partei $P \in \{A, B\}$:

Objekt O_P: Die Signatur $sig_P(\text{Vertrag})$ oder die Ersatzsignatur $sig_V(sig_P(TID, \text{GenInfo}, \text{Vertrag}))$ des Vermittlers über die Generierungsinformation für O_P.

Überprüfen von O_P: Mit dem öffentlichen Schlüssel von P und dem Vertragstext wird die Gültigkeit der Signatur $sig_P(\text{Vertrag})$ überprüft. Falls es sich um eine Ersatzsignatur handelt, muß zuerst die Signatur $sig_P(TID, \text{GenInfo}, \text{Vertrag})$ verifiziert werden. Dabei ist wesentlich, daß der richtige Vertragstext verwendet wurde und daß diese Signatur als Generierungsinformation dient (hier durch „GenInfo“ gekennzeichnet). Auch die verwendete Transaktionsnummer muß zur aktuellen Transaktion passen. Schließlich muß noch die Bestätigung des Vermittlers geprüft werden, indem man die Signatur $sig_V(sig_P(TID, \text{GenInfo}, \text{Vertrag}))$ mit dem öffentlichen Schlüssel des Vermittlers verifiziert.

Generierungsinformation für O_P: Die Signatur $sig_P(TID, \text{GenInfo}, \text{Vertrag})$ gibt dem Vermittler die Erlaubnis, falls nötig, einen Ersatzvertrag zu generieren.

Überprüfen der Generierungsinformation für O_P: Die Signatur $sig_P(TID, \text{GenInfo}, \text{Vertrag})$ muß mit dem öffentlichen Schlüssel der Partei P verifiziert werden. Dabei ist wesentlich, daß die aktuelle Transaktionsnummer verwendet wurde, daß diese Signatur als Generierungsinformation dient und daß der richtige Vertragstext signiert wurde. Wenn alle diese Bedingungen erfüllt sind, garantiert dies die Generierbarkeit durch den Vermittler.

Generieren von O_P: Der Vermittler bekräftigt mit seiner Signatur $sig_V(sig_P(TID, \text{GenInfo}, \text{Vertrag}))$ über die Generierungsinformation, daß er diesen Vertrag für gültig erklärt.

Aufgrund der starken Generierbarkeit beider Objekte können sowohl aktive als auch optimistische Protokolle verwendet werden. Da keine Nichtabstreitbarkeit des Empfangs garantiert werden muß, kann der Austausch mit den Protokollen P1 oder P2 durchgeführt werden. Nichtabstreitbarkeit der Herkunft wird hier nicht benötigt, da sich diese Eigenschaft anhand der Signaturen sowieso überprüfen läßt.

Die hier vorgeschlagene Modellierung realisiert eine Vertragsunterzeichnung mit den geforderten Eigenschaften. Eine dritte Partei, die einen auf diese Weise ausgetauschten Vertrag überprüfen will, muß lediglich die unter „Überprüfen von O_P“ beschriebenen Schritte durchführen. Dabei ist die verwendete Transaktionsnummer für die Gültigkeit des Vertrags irrelevant, da sich der genaue Kontext sowieso aus dem Vertragstext heraus ergeben sollte. Dagegen darf die als Generierungsinformation geleistete Unterschrift $sig_P(TID, \text{GenInfo}, \text{Vertrag})$ für sich allein genommen keine Bedeutung außerhalb des Austauschprotokolls haben. Insbesondere kann sie nicht als Zustimmung der Partei P zum Vertrag gewertet werden, da der Austausch auch abgebrochen worden sein könnte.

Durch die Fairneßeigenschaften des verwendeten Protokolls ist schließlich immer sichergestellt, daß entweder beide Parteien den unterschriebenen Vertrag erhalten oder daß keine Partei an den Vertrag gebunden ist.

Implementierung

Die angegebene Modellierung der Vertragsunterzeichnung habe ich im Rahmen der Beispielanwendung `ContractSigning` mit FlexiFair implementiert. Die Objekte und Beschreibungen lassen sich präzise auf entsprechende Klassen abbilden. Diese Abbildung sieht wie folgt aus:

Objekt O_P: Die Klasse `GenContract` dient als Container für die Signatur von P bzw. die Ersatzsignatur vom Vermittler.

Überprüfen von O_P: Die Beschreibung des Objekts O_P wird bei der Implementierung durch die Klasse `GenContractDescription` realisiert. Die Methode `verifyItem` aus dieser Klasse führt die angegebenen Überprüfungen durch.

Generierungsinformation für O_P: Diese Information ist in Java ein einfacher Byte-Array, der die Signatur $sig_P(TID, \text{GenInfo}, \text{Vertrag})$ enthält.

Überprüfen der Generierungsinformation für O_P: Die Methode `verifyGenInfo` in der Klasse `GenContractDescription` führt diese Überprüfung wie oben beschrieben durch.

Generieren von O_P: Der Vermittler generiert das Objekt O_P, indem er die Methode `generateItemByTTP` der Klasse `GenContract` aufruft. Der Vermittler muß dazu die Generierungsinformation, die Beschreibung von O_P, die Transaktionsnummer und

eine geheime Zusatzinformation kennen. Bei dieser Zusatzinformation handelt es sich um die Klasse `ContractGenerateSecret`, mit der die Ersatzsignaturen des Vermittlers erstellt werden können.

4.7.2 Empfangsbestätigung für eine E-Mail

Beim Austausch einer E-Mail gegen eine Empfangsbestätigung handelt es sich um die elektronische Form eines Einschreibens mit Rückschein. Der Absender einer Nachricht möchte sicherstellen, daß der Empfänger diese Nachricht erhalten hat bzw. noch erhalten kann und daß sich dies gegenüber anderen Parteien beweisen läßt. Die Nachricht darf deswegen nur genau dann veröffentlicht werden, wenn deren Empfang quittiert wird.

Bei dieser Art des fairen Austausches wird also ein beliebiger vorher unbekannter Text gegen eine Signatur des Empfängers ausgetauscht, die sich auf diesen Text bezieht. Daraus ergeben sich Anforderungen an den Austausch, die sich deutlich von anderen Szenarien wie z.B. Vertragsunterzeichnung unterscheiden:

- Es wird eine Beschreibung für einen beliebigen vorher unbekannten Text benötigt. Dieser Text muß während der Austauschtransaktion unveränderlich sein, d.h. unabhängig davon, ob ein Fehler aufgetreten ist oder nicht, muß nachprüfbar immer der gleiche Text bekanntgegeben werden.
- Die Empfangsbestätigung muß sich auf den ausgetauschten Text beziehen, obwohl dieser nur bei einem erfolgreichen Austausch verraten werden darf.

Gemäß der ersten Bedingung ist der ausgetauschten Text frei wählbar und muß trotzdem unveränderlich festgelegt sein. Um diesen Widerspruch aufzulösen, wird nicht der Text ausgetauscht, sondern nur der symmetrische Schlüssel key, mit dem man den vorher festgelegten verschlüsselten Text $VT = e_{key}(\text{Text})$ entschlüsseln kann.

Die zweite Bedingung definiert eine Abhängigkeit zwischen den ausgetauschten Objekten, was folgendes Problem aufwirft: Da FlexiFair das Ziel hat, aktive und optimistische Protokolle gleichermaßen zu unterstützen, ohne daß deren Unterschiede von der Anwendung berücksichtigt werden müssen, dürfen keine Abhängigkeiten zwischen den Objekten existieren. Insbesondere bei aktiven Protokollen müssen beide Parteien ihre Objekte an den Vermittler liefern, ohne daß das Objekt der anderen Partei einen Einfluß auf die Erzeugung des eigenen Objekts haben kann. Dieser Widerspruch zur zweiten Bedingung kann durch den Einsatz von Protokollen mit Nachweis des Empfangs aufgelöst werden. Die Empfangsbestätigung für den gelieferten Text besteht daher nur aus einem Nachweis des Empfangs, der sowieso immer abhängig vom empfangenen Objekt ist.

Nach diesen Vorüberlegungen folgt jetzt die genaue Modellierung der Objekte und ihrer Beschreibungen, sowie die Anforderungen an die verwendeten Fairneßprotokolle. Zuerst müssen jedoch einige bestimmte Informationen schon nach dem Aushandeln in Modul M1 bekannt sein:

Öffentliche Information: Der Hashwert $HW = h(key)$ des symmetrischen Schlüssels und der verschlüsselte Text $VT = e_{key}(\text{Text})$.

Mit Hilfe dieser Werte können die beiden ausgetauschten Objekte als generierbare Objekte modelliert werden:

Objekt O_A: Der symmetrische Schlüssel key und der verschlüsselte Text $e_{key}(\text{Text})$. Obwohl der verschlüsselte Text bereits bekannt ist, ist dieser trotzdem im Objekt O_A enthalten, um dessen Überprüfbarkeit zu vereinfachen.

Überprüfen von O_A: Der Hashwert des von A gelieferten Wertes key muß gleich dem vorgegebenen Hashwert HW sein. Außerdem wird der gelieferte verschlüsselte Text mit VT verglichen.

Generierungsinformation für O_A: Es wird $E_V(TID, key)$ und $e_{key}(\text{Text})$ geliefert, sowie die Signatur von A für diese Werte. Da der Empfänger dieser Generierungsinformation lediglich diese Signatur $sig_A(TID, \text{GenInfo}, E_V(TID, key), e_{key}(\text{Text}))$ überprüfen kann, ohne etwas über die Korrektheit der Verschlüsselung $E_V(TID, key)$ zu erfahren, wird auf diese Weise nur schwache Fairneß garantiert (siehe auch das Beispiel in Abschnitt 4.4.1).

Überprüfen der Generierungsinformation für O_A: Da O_A nur schwach generierbar ist, genügt die Überprüfung der Signatur $sig_A(TID, \text{GenInfo}, E_V(TID, key), e_{key}(\text{Text}))$. Dies schließt ein, daß die Korrektheit der verwendeten Werte TID, GenInfo und $e_{key}(\text{Text})$ geprüft wird.

Generieren von O_A: Der Vermittler entschlüsselt $E_V(TID, key)$ und erhält so den Schlüssel key. Außerdem stellt der Vermittler noch den verschlüsselten Text $e_{key}(\text{Text})$ bereit. Das Generieren kann fehlschlagen, wenn der verschlüsselte Wert die falsche Transaktionsnummer enthält oder wenn sich das generierte O_A bei der Überprüfung als fehlerhaft herausstellt.

Objekt O_B: Das leere Objekt.

Überprüfen von O_B: Die Überprüfung ist immer erfolgreich.

Generierungsinformation für O_B: Eine Signatur $sig_B(TID, \text{GenInfo})$ von B, die dem Vermittler erlaubt, das leere Objekt zu generieren. Da sich das leere Objekt trivialerweise immer generieren läßt, wird starke Generierbarkeit für O_B garantiert.

Überprüfen der Generierungsinformation für O_B: O_B ist dann stark generierbar, wenn die Signatur $sig_B(TID, \text{GenInfo})$ gültig ist.

Generieren von O_B: Der Vermittler hat hier nichts zu tun.

Die Objekte besitzen also starke und schwache Generierbarkeit, so daß als Austauschprotokoll wahlweise ein aktives oder optimistisches Protokoll in Frage kommt. Es muß jedoch immer ein Nachweis des Empfangs für das Objekt O_A erzeugt werden, so daß die Protokolle P1-NE und P2-NE besonders gut geeignet sind. Der Nachweis des Empfangs für das Objekt O_A sieht dann wie folgt aus:

Nachweis des Empfangs für O_A: Eine Signatur $sig_B(TID, \text{NE}, O_A)$ von B oder die vom Vermittler generierte Ersatzsignatur $sig_V(TID, \text{NE-B}, O_A)$ dienen als Nachweis des Empfangs durch B (NE-B). Nur in optimistischen Protokollen kann die Signatur von B selbst erstellt werden. Bei der Konfliktlösung durch den Vermittler oder bei aktiven Protokollen muß immer der Vermittler die Signatur erstellen.

Es bleibt zu zeigen, daß auf diese Weise wirklich die Funktion einer Empfangsbestätigung für einen vorher unbekannten Text realisiert wird. Die Partei B erhält bei einem erfolgreichen Austausch key und $e_{key}(\text{Text})$, woraus sie den gesendeten Text berechnet. Da ein beliebiger Text ausgetauscht werden kann, sollte es sich dabei am besten um einen von A signierten Text handeln. Dann kann sich B sofort nach dem Entschlüsseln davon überzeugen, daß der entschlüsselte Text sinnvoll ist.

Auf der anderen Seite erhält A den Nachweis des Empfangs für die Werte key und $e_{key}(\text{Text})$, womit er beweisen kann, daß B den Text in dieser Austauschtransaktion erhalten hat oder noch erhalten kann. Das Nachprüfen der Empfangsbestätigung durch zum Beispiel eine dritte Partei C geschieht dann wie folgt:

Die Partei A muß $sig_B(TID, \text{NE}, O_A)$ oder $sig_V(TID, \text{NE-B}, O_A)$, sowie die Transaktionsnummer TID und $O_A = key, e_{key}(\text{Text})$ vorlegen. Außerdem sagt A dem Dritten C, welche Partei B diese Empfangsbestätigung geliefert hat.

Die Partei C überprüft zuerst die Signatur $sig_B(TID, \text{NE}, O_A)$ mit dem öffentlichen Schlüssel von B. Falls $sig_V(TID, \text{NE-B}, O_A)$ geliefert wurde, muß er die Signatur mit dem öffentlichen Schlüssel des Vermittlers prüfen. Wenn die Signatur gültig ist, entschlüsselt C mit key den verschlüsselten Text $e_{key}(\text{Text})$. Auf diese Weise hat sich C davon überzeugt, daß B in der Austauschtransaktion mit der Nummer TID diesen Text erhalten hat bzw. noch vom Vermittler erhalten kann.

In der Literatur (z.B. [ZDB00, KM01]) wird bei der Empfangsbestätigung für eine E-Mail manchmal noch als weitere Protokolleigenschaft gefordert, daß der Vermittler den ausgetauschten Text nicht lernen soll. Diese Bedingung kann mit der hier vorgestellten Modellierung leicht erfüllt werden, wenn überall der verschlüsselte Text $e_{key}(\text{Text})$ durch seinen Hashwert $H(e_{key}(\text{Text}))$ ersetzt wird. Dann lernt der Vermittler den verschlüsselten Text nicht mehr, so daß der Inhalt geheim bleibt. Beim Überprüfen der Empfangsbestätigung läßt sich anhand des Hashwerts eindeutig der richtige verschlüsselte Text zuordnen, wodurch die Aussagekraft der Empfangsbestätigung auch in diesem Fall gewährleistet ist.

4.7.3 Austausch Geld gegen Ware

Das Bezahlen im Internet stellt ein typisches Anwendungsszenario für den Fairneßdienst FlexiFair dar. Wenn ein Kunde den elektronischen Einkauf einer digitale Ware über das Internet abwickelt, möchte er sichergehen, daß der Austausch von Geld gegen Ware fair abläuft. Dieser elektronische Einkauf stellt aufgrund der Verschiedenartigkeit der ausgetauschten Objekte eine besondere Herausforderung für die Flexibilität von FlexiFair dar. Auf der einen Seite gibt es die Ware, die ein beliebiges digitales Objekt sein kann. Auf der anderen Seite gibt es das Geld, dessen Gültigkeit bei elektronischen online

Zahlungssystemen [Cha83, Sch97, KV01a, KV01b] jedoch nur von der Bank vollständig überprüft werden kann, da nur die Bank das mehrfache Ausgeben einer Münze (auch double-spending genannt) erkennen kann.

Eine weitere Anforderung an FlexiFair stellt die mögliche Anonymität des Kunden bei elektronischen Zahlungen dar. Der Einsatz eines anonymen Zahlungssystems macht nur dann Sinn, wenn der Kunde auch bei der Konfliktlösung anonym bleiben kann. Das eingesetzte Fairneßprotokoll darf also nicht die Identität des Kunden preisgeben, um Fairneß zu erreichen. FlexiFair unterstützt grundsätzlich diese Anforderung, da alle Parteien den Austausch unter einem selbstgewählten Pseudonym durchführen können. Bei Protokollen mit F4-Fairneß und T2-Terminierung besteht dann kein Grund zur Deanonymisierung.

Die Anonymität des Kunden wird im folgenden aber nicht weiter betrachtet, da nur sehr wenige Zahlungssysteme das anonyme Widerrufen von Zahlungen unterstützen. Gerade bei Zahlungssystemen, die die durch die Anonymität gegebenen Mißbrauchsmöglichkeiten minimieren (wie z.B. [CMS96, KV01d, Küg02]), besteht noch Forschungsbedarf zum Thema anonyme Rückabwicklung von Zahlungen. Stattdessen beschränke ich mich hier auf ein einfaches Scheck-ähnliches Zahlungssystem. Der Kunde stellt dabei einfach eine Signatur über den geforderten Betrag aus und die Bank transferiert diesen Betrag vom Konto des Kunden auf das Konto des Verkäufers.

Im folgenden wird der Kunde die Rolle der Partei A einnehmen, während der Verkäufer mit B bezeichnet wird. Dann ist das Objekt O_A eine widerrufbare Zahlung und O_B die zu liefernde Ware.

Öffentliche Information: Der zu zahlende Betrag und die Beschreibung der Ware.

Objekt O_A: Die Zahlung mittels der Signatur $sig_A(TID, \text{Zahlung von } A \text{ an } B, Betrag)$. Erst wenn diese Signatur bei der Bank eingereicht wird, erhält B den Betrag gutgeschrieben.

Überprüfen von O_A: Es muß die Gültigkeit der Signatur $sig_A(TID, \text{Zahlung von } A \text{ an } B, Betrag)$ nachgeprüft werden. Da die Transaktionsnummer eindeutig ist, kann es sich nicht um das Wiedereinspielen einer alten Zahlung handeln, so daß die Bank beim erstmaligen Erhalt dieser Signatur den angegebenen Betrag vom Konto des Kunden A auf das Konto des Händlers B überweist.

Widerrufen von O_A: Wenn sich der Vermittler an die Bank wendet und das Widerrufen der Zahlung aus der Transaktion TID fordert, wird die Bank den entsprechenden Betrag auf das Konto des Kunden zurückbuchen. Falls noch keine Buchung unter dieser Transaktionsnummer erfolgt ist, muß nichts zurückgebucht werden. Grundsätzlich wird die Bank nach einem Widerruf keine Einreichungen mehr unter dieser Transaktionsnummer akzeptieren, wenn es sich um einen Geldtransfer von A an B handelt.

Objekt O_B: Eine beliebige Ware, die sich durch einen Bitstring darstellen läßt. Mögliche Beispiele sind eine Bild-, Video- oder Musikdatei oder auch Kino- oder Konzertkarten.

Überprüfen von O_B: Es muß eine hinreichend genaue Überprüfung der Ware möglich sein. Zum Beispiel kann der Hashwert einer Musikdatei als Beschreibung vorliegen, gegen die geprüft wird. Bei Kino- oder Konzertkarten handelt es sich im wesentlichen um digitale Signaturen, die einfach zu verifizieren sind.

Generierungsinformation für O_B: Es wird die für den Vermittler verschlüsselte Ware $E_V(TID, Ware)$ zusammen mit der Signatur $sig_B(TID, \text{GenInfo}, E_V(TID, Ware))$ benötigt.

Überprüfen der Generierungsinformation für O_B: Da O_B nur schwach generierbar ist, genügt die Überprüfung der Signatur $sig_B(TID, \text{GenInfo}, E_V(TID, Ware))$.

Generieren von O_B: Der Vermittler entschlüsselt $E_V(TID, Ware)$ und erhält so die Ware. Das Generieren kann fehlschlagen, wenn der verschlüsselte Wert die falsche Transaktionsnummer enthält oder wenn die entschlüsselte Ware nicht der Beschreibung entspricht.

Die hier beschriebenen Eigenschaften der Objekte erlauben sowohl ein aktives Protokoll wie P1 als auch ein optimistisches Protokoll wie P3. Falls die Ware noch zusätzliche Eigenschaften wie starke Generierbarkeit oder schwache Widerrufbarkeit besitzt, sind auch die Protokolle P4 oder P5-NE für diesen Austausch geeignet.

4.7.4 Simultanes Aufdecken von Pseudonymen

Wenn zwei pseudonyme Parteien, die über das Internet miteinander kommunizieren, sich darauf verständigen, ihre hinter den Pseudonymen verborgenen Identitäten preiszugeben, entsteht auch ein Fairneßproblem: Die Partei die zuerst die Identität der anderen erfährt, könnte die Kommunikation abbrechen und die eigene Identität weiterhin geheimhalten. Ebenso könnte eine Partei einen frei erfundenen Namen liefern und sich zum Beispiel als Micky Maus ausgeben.

Es soll also auf faire Weise festgestellt werden, welche Identität P' sich hinter einem bisher verwendeten Pseudonym $P \in \{A, B\}$ versteckt. Dazu muß einerseits eine vertrauenswürdige Partei die Identität P' verifiziert haben und diese bestätigen. Andererseits muß nachgewiesen werden, daß es sich bei P und P' um dieselbe Person handelt.

Gerade bei einem derart speziellen Austauschvorgang kann der generische Fairneßdienst FlexiFair seine Vorteile gegenüber weniger vielseitigen Lösungen ausspielen. Obwohl diese neue Einsatzmöglichkeit von fairem Austausch in der Literatur noch nicht diskutiert wurde, kann FlexiFair ohne irgendwelche Anpassungen sofort eine Lösung für dieses Problem bereitstellen. Es handelt sich also um ein Beispiel, wie eine Anwendung bei einem neuen, vorher unbekannten Austauschproblem auf FlexiFair zurückgreift und so den fairen Austausch einfach und sicher realisiert.

Im folgenden wird zuerst die Modellierung der Objekte vorgestellt. Da beide Parteien die gleiche Art von Objekten verwenden, genügen diese Ausführungen für die Partei P. Anschließend wird das verwendete Austauschprotokoll beschrieben, mit dem der Austausch der Identitäten vollzogen werden kann.

Objekt O_P: Die wahre Identität P' von Partei P, ein Zertifikat von einer Zertifizierungsinstanz über einen Signaturschlüssel von P' und eine Signatur $sig_{P'}(TID, \text{Ident})$ mit diesem Signaturschlüssel.

Überprüfen von O_P: Zuerst einmal muß das Zertifikat der Zertifizierungsinstanz für den Signaturschlüssel von P' gültig sein. Ebenso muß die mit diesem Schlüssel erstellte Signatur $sig_{P'}(TID, \text{Ident})$ korrekt sein. Da die signierte Transaktionsnummer bei jedem Austausch verschieden ist, kann die überprüfende Partei sicher sein, daß es sich um eine neue Signatur von P' handelt. Außerdem wird zur Berechnung von TID auch der öffentliche Schlüssel von P verwendet, so daß der Zusammenhang zwischen dem Pseudonym P und P' nachgewiesen ist.

Da die ausgetauschten Objekte keine Generierbarkeit und keine Widerrufbarkeit besitzen, kommen für den Austausch nur aktive Protokolle wie z.B. P1 in Frage. Ein weiterer Grund für die Beschränkung auf aktive Protokolle besteht darin, daß die Beschreibung dieser Objekte dem Vermittler bisher unbekannt ist. Bei aktiven Protokollen kann diese Beschreibungen zur Laufzeit mitgeliefert werden, so daß der Vermittler einfach um die benötigte Funktionalität erweitert wird. Schließlich wird der Vermittler beim Einsatz von P1 die Objekte (und damit die Identitäten von beiden Parteien) nur dann austauschen, wenn beide Objekte einer Überprüfung standhalten. Auf diese Weise werden beide Parteien fair behandelt, und es werden entweder die Identitäten von beiden Parteien aufgedeckt oder von keiner.

4.8 Zusammenfassung

In diesem Kapitel habe ich verschiedene Probleme behandelt, die bei der Implementierung von FlexiFair zu berücksichtigen waren.

Die Sicherheit und Korrektheit von fairem Austausch kann nur durch die Verwendung einer eindeutigen Transaktionsnummer sichergestellt werden. Durch die Berechnung dieser Transaktionsnummer aus verschiedenen Protokollparametern kann ich auf einfache Weise Verbindungen zwischen verschiedenen Nachrichten und Daten einer Transaktion herstellen. Diese Zusammenhänge können dann auch durch andere Parteien überprüft werden, um sich von der Korrektheit von Nachrichten zu überzeugen.

Die Beschreibung von Objekten ist bei FlexiFair nicht auf wenige vordefinierte Klassen beschränkt, sondern kann mit dem in Abschnitt 4.3.2 vorgeschlagenen dynamischen Ansatz auch neu entwickelte Beschreibungen verwenden. Auf diese Weise kann jedes digitale Objekt mit FlexiFair ausgetauscht werden.

Die in Abschnitt 4.4.1 vorgestellte Implementierung der auszutauschenden Objekte erlaubt die Modellierung von Generierbarkeit und Widerrufbarkeit als Erweiterung eines digitalen Objekts. Auch die Nachweise der Herkunft und des Empfangs werden als besondere digitale Objekte aufgefaßt, wobei für den Nachweis des Empfangs jedoch die Unterstützung durch ein geeignetes Protokoll benötigt wird.

Die Beispiele in Abschnitt 4.7 beschreiben für verschiedene Szenarien eine mögliche Implementierung unter Verwendung von FlexiFair. Dabei zeige ich, daß FlexiFair neben

den bekannten Austauschproblemen wie Vertragsunterzeichnung, Empfangsbestätigung für eine E-Mail und faires Einkaufen auch Austauschprobleme in bisher unbekannten Szenarien lösen kann. Das Beispiel des simultanen Aufdeckens von Identitäten demonstriert diese Eigenschaft von FlexiFair, für jede Anwendung geeignete Austauschmechanismen zur Verfügung stellen zu können.

5 Sichere Hardware zur Unterstützung von fairem Austausch

In den vorherigen Kapiteln wurde stets der Fall betrachtet, daß ein Austausch zwischen zwei Parteien ausgeführt wird, die sich gegenseitig nicht vertrauen und über unzuverlässige Verbindungen kommunizieren. In diesem Kapitel wird untersucht, wie mittels sicherer Hardware (z.B. Chipkarten) das Vertrauen zwischen den Parteien erhöht werden kann, was dann zur Vereinfachung und Verbesserung von Austauschprotokollen genutzt wird.

Durch die Ausführung von Fairneßprotokollen innerhalb der sicheren Hardware hat ein Angreifer deutlich weniger Einfluß auf den Protokollablauf, was seine Betrugsmöglichkeiten reduziert. Da die Kommunikationsverbindung auch in diesem Szenario von einem Angreifer gekappt werden kann, läßt sich fairer Austausch nicht allein durch den Einsatz von sicherer Hardware garantieren. Stattdessen ist man immer noch auf die Hilfe eines ständig erreichbaren Vermittlers zur Konfliktlösung angewiesen.

Das Ziel dieses Kapitels ist es, das Potential von hardwareunterstützten Fairneßprotokollen zu untersuchen. Es wird gezeigt, daß der Einsatz von sicherer Hardware deutliche Vorteile gegenüber in Software implementierten optimistischen und aktiven Austauschprotokollen besitzt. Die hier vorgeschlagenen Protokolle sind besonders effizient und vielfältig einsetzbar. Insbesondere der in Abschnitt 5.4 vorgestellte Austausch von verderblichen Waren scheint ohne Hardware unmöglich zu sein.

Einen Teil dieser hier beschriebenen Ergebnisse habe ich bereits in [VPG01, VGP03, PVG02] veröffentlicht. In der Literatur wurde diese Nutzung von sicherer Hardware für fairen Austausch ansonsten noch nicht untersucht. In [ZL99] wird zwar sichere Hardware beim Austausch von Geld gegen Ware verwendet, aber Fairneß kann dieser Ansatz trotzdem nicht garantieren.

Im folgenden werden zuerst die Eigenschaften der sicheren Hardware und ihre Einsatzmöglichkeiten vorgestellt. In Abschnitt 5.2 beschreibe ich ein optimistisches Austauschprotokoll, das von einer Chipkarte als sichere Hardware unterstützt wird, so daß fairer Austausch beliebiger digitaler Objekte möglich ist. Im danach folgenden Abschnitt schlage ich Konzepte vor, mit denen sich der manchmal recht hohe Rechen- und Speicheraufwand beim fairen Austausch so steuern läßt, daß eine leistungsschwache Chipkarte die vorgestellten Protokolle trotzdem bewältigen kann. Lösungen für den fairen Austausch von verderblichen Waren präsentiere ich dann in Abschnitt 5.4. Zum Schluß entwerfe ich in Abschnitt 5.5 ein generisches Protokoll, daß die Vorteile von optimistischen Protokollen mit denen von aktiven Protokollen vereint und dadurch F4-Fairneß mit T2-Terminierung für beide Parteien erzielt. Damit beantworte ich die offene Frage von Asokan [Aso98, S.137], ob ein optimistisch fairer Austausch von beliebigen Objekten möglich ist. Ein

weiterer Vorteil dieses Protokolls besteht darin, daß es im Bezug auf die Anzahl der ausgetauschten Nachrichten noch effizienter als bisher bekannte Protokolle ist.

5.1 Sichere Hardware

Zuerst wird das im folgenden verwendete Szenario für den Einsatz von sicherer Hardware vorgestellt. Dann definiere ich die Anforderungen, die von sicherer Hardware erfüllt werden müssen, um fairen Austausch zu unterstützen. Danach diskutiere ich noch kurz, welche Auswirkungen sichere Hardware auf die Lösbarkeit von fairem Austausch hat.

5.1.1 Systemannahmen

Die Grundidee beim Einsatz von sicherer Hardware besteht darin, daß diese das Austauschprotokoll für ihren Besitzer ausführt und dabei immer gemäß dem verwendeten Protokoll handelt. Die Verfügbarkeit der sicheren Hardware kann aber nur für die Partei sichergestellt werden, die im Besitz der Hardware ist und lokal mit ihr kommunizieren kann.

Man kann nun zwei verschiedene Szenarien bei Einsatz von sicherer Hardware unterscheiden:

Eine Partei besitzt sichere Hardware: Die Partei mit der sicheren Hardware hat den Vorteil, daß sie lokal mit der Hardware kommunizieren kann und dabei keine Verbindungsabbrüche befürchten muß. Die andere Partei besitzt die Gewißheit, daß das Protokoll vom Hardwaregerät korrekt ausgeführt wird. Die Kommunikation mit der Hardware kann jedoch einfach unterbrochen werden.

Beide Parteien besitzen sichere Hardware: Die Kommunikation mit der jeweils entfernten Partei ist auch hier durch Verbindungsabbrüche bedroht. Es scheint also keinen wirklichen Vorteil durch den Einsatz von zwei Geräten zu geben, so daß diese Variante in dieser Arbeit nicht weiter betrachtet wird.

Im folgenden werden also nur Protokolle untersucht, die mit einem einzigen Hardwaregerät auskommen. Um dieses Szenario etwas konkreter zu gestalten, verwende ich einige Begriffe, die sich an dem Beispiel orientieren, daß ein Kunde mit einer Chipkarte einen Einkauf bei einem Händler durchführen will: Die Partei, die sich im Besitz der sicheren Hardware befindet, nenne ich den *Kunden* K und die andere Partei *Händler* H. Das Hardwaregerät des Kunden wird dann auch als *Chipkarte* C bezeichnet. Dies schließt jedoch nicht aus, daß die sichere Hardware z.B. auch als Einsteckkarte für den Computer des Kunden realisiert werden kann. Die Chipkarte ist mit dem Rechner des Kunden verbunden und kommuniziert ausschließlich über diesen (siehe auch Abbildung 5.1). Der Kunde kontrolliert also jegliche Kommunikation mit der Chipkarte. Nachrichten des Händlers an die Chipkarte müssen daher vom Kunden weitergeleitet werden, ebenso wie eine Antwort der Chipkarte an den Händler.

Die ausgetauschten Objekte nenne ich O_K und O_H, wobei O_K das Objekt des Kunden und O_H das des Händlers ist. Es existieren Beschreibungen für beide Objekte, und

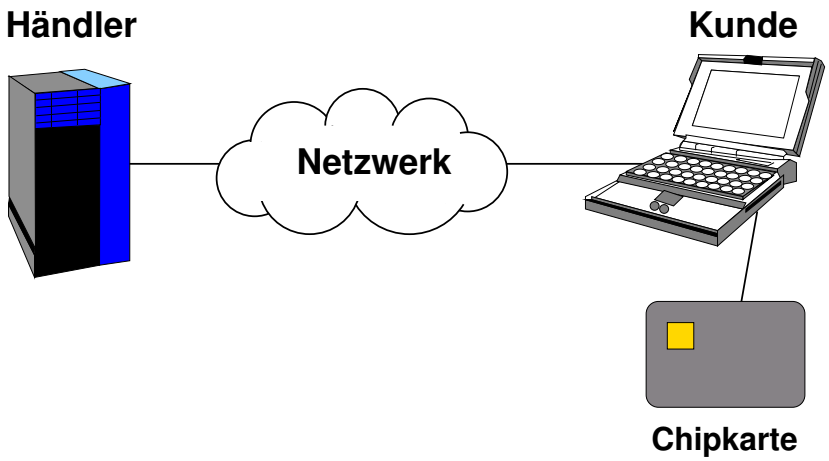

Abbildung 5.1: Das Szenario für fairen Austausch mit sicherer Hardware.

die Chipkarte kann mit diesen Beschreibungen die Korrektheit der erhaltenen Objekte überprüfen.

5.1.2 Hardwareanforderungen

Die Bezeichnung *sichere Hardware* bzw. *Chipkarte* wird im folgenden für Geräte verwendet, die selbständig interaktive Protokolle ausführen können und die folgenden Eigenschaften besitzen:

- Manipulationssicherheit
- Vertrauen sowie Authentizität, Integrität und Vertraulichkeit von Nachrichten
- Verläßliche Zustandsinformationen

Diese Eigenschaften werden im folgenden kurz erläutert:

Manipulationssicherheit

Eine sichere Hardware muß gegen das Auslesen oder Verändern von geheimen Daten geschützt sein. Außerdem muß ein gestartetes Austauschprotokoll immer exakt nach dessen Spezifikationen abgearbeitet werden, d.h. ein Angreifer kann den Protokollablauf nicht manipulieren. Diese Anforderungen an die Manipulationssicherheit der Hardware müssen auch dann erfüllt sein, wenn ein Angreifer direkten Zugriff auf das Gerät hat und vor dessen Zerstörung nicht zurückschreckt.

Vertrauen sowie Authentizität, Integrität und Vertraulichkeit von Nachrichten

Die in Hardware realisierten Austauschprotokolle müssen den Spezifikationen entsprechen und dürfen keine undokumentierten Funktionen enthalten. Da sich dies nur schwer vom

Besitzer eines solchen Geräts nachprüfen läßt, müssen der Hersteller und der Herausgeber einer solchen Hardware vertrauenswürdig sein bzw. durch Zertifizierung die fehlerfreie, spezifikationskonforme Funktion der Hardware nachweisen.

Zusätzlich muß auch jemand, der nicht selbst im Besitz der Hardware ist und nur über ein Netzwerk an einer Protokollausführung beteiligt ist, die Authentizität von erhaltenen Nachrichten prüfen können. Deshalb muß in der Hardware ein geheimer Signaturschlüssel gespeichert sein, mit dem alle sicherheitsrelevanten Nachrichten unterschrieben werden. Der entsprechende öffentliche Schlüssel muß von einer vertrauenswürdigen Partei, z.B. dem Herausgeber der Hardware, zertifiziert sein, um die Zugehörigkeit des Signaturschlüssels zur sicheren Hardware überprüfbar zu machen.

In umgekehrter Weise muß auch die Hardware in der Lage sein, die Authentizität von empfangenen Nachrichten anhand von digitalen Signaturen zu überprüfen. Dafür muß in der Hardware bereits ein öffentlicher Schlüssel einer Zertifizierungsinstanz gespeichert sein, mit dem Zertifikate von Kommunikationspartnern verifiziert werden können. Daher kann ein Angreifer keine fremde Identität vortäuschen.

Bei der Kommunikation zwischen den beteiligten Parteien und der Chipkarte soll auch die Vertraulichkeit von Nachrichten gewährleistet sein. Daher muß jede Partei einen symmetrischen Schlüssel mit der Chipkarte vereinbaren, um ein Abhören der damit verschlüsselten Nachrichten zu unterbinden.

Zusätzlich zum Signieren und Verschlüsseln von Nachrichten müssen wieder alle Nachrichten die Transaktionsnummer enthalten (siehe Abschnitt 4.1), wodurch ein Angriff wie das Wiedereinspielen einer alten Nachricht erkannt und somit abgewehrt wird.

Wegen diesen Maßnahmen zur Sicherung der Kommunikation kann ein Angreifer lediglich Nachrichten verzögern oder löschen. Außerdem kann ein Angreifer Daten bei den beteiligten Parteien löschen, falls diese Parteien mit dem Angreifer kooperieren. Auch die Zerstörung der Chipkarte ist mit Hilfe des Kunden möglich, was deren Daten löscht und jegliche weitere Kommunikation unterbindet.

Verläßliche Zustandsinformationen

Die Hardware muß mit signierten Nachrichten Auskunft über ihren aktuellen Zustand geben können. Ein solcher Mechanismus stellt eine verläßliche Informationsquelle dar, die insbesondere nach einer Kommunikationsunterbrechung die Fortsetzung eines Protokolls an der richtigen Stelle ermöglicht. Eine Voraussetzung für verläßliche Zustandsinformationen ist die persistente Speicherung des Hardwarezustands und der gerade verarbeiteten Daten. Eine Unterbrechung der Stromversorgung darf also nie zum Verlust von Informationen führen. Der Zustand muß daher in der Hardware selbst gespeichert werden. Die externe Speicherung des Zustands ähnlich den Konzepten in Abschnitt 5.3.1 ist dagegen nicht möglich, weil ein Angreifer sonst Manipulationen am Zustand des Austausches vornehmen könnte.

5.1.3 Der Schwierigkeitsgrad von fairem Austausch mit sicherer Hardware

Die zusätzlichen, durch die sichere Hardware bereitgestellten Garantien vereinfachen den fairen Austausch deutlich, wodurch im folgenden verschiedene neue Protokolle konstruiert werden können. Eine Chipkarte kann einzelne Aufgaben des Vermittlers (hier auch *externer Vermittler* genannt) übernehmen und ohne fremde Hilfe selbst ausführen. Daher könnte man die sichere Hardware auch als einen *lokalen Vermittler* bezeichnen und die Protokolle können als eine Mischung von aktivem und optimistischen Austausch angesehen werden.

Im Vergleich zum externen Vermittler fehlt jedoch der sicheren Hardware eine entscheidende Eigenschaft: Sie ist nur für ihren Besitzer ständig verfügbar. Der Händler kann sich zur Konfliktlösung jedoch nicht immer an die Chipkarte wenden, da der Kunde die Kommunikation blockieren kann. Aus diesem Grund behalten die Beweise für die Unmöglichkeit eines fairen Austausches ohne Vermittler [EY80, PG99] weiterhin ihre Gültigkeit. Auch mit sicherer Hardware kann nicht auf den Vermittler verzichtet werden, um Fairneß für beide Parteien zu garantieren.

5.2 Ein Austauschprotokoll mit sicherer Hardware

In diesem Abschnitt stelle ich ein einfaches Protokoll für stark fairen Austausch mit Unterstützung von sicherer Hardware vor. Es handelt sich um ein optimistisches Protokoll, das die Unterstützung eines externen Vermittlers nur in wenigen Fehlerfällen benötigt, und es ermöglicht den Austausch von beliebigen digitalen Objekten. Im Gegensatz dazu ist bei in Software implementierten optimistischen Protokollen keine Lösung bekannt, wie beliebige digitale Objekte ohne Generierbarkeit und Widerrufbarkeit Fairneß garantieren können (siehe auch Abschnitt 3.1 oder [Aso98, am Ende von Abschnitt 2.3.1]).

Diese neuen Protokolleigenschaften zeigen, daß es sich bei einem Austauschprotokoll mit Chipkartenunterstützung um eine Synthese aus einem Protokoll mit aktivem Vermittler und einem optimistischen Protokoll handelt. Genau wie bei aktiven Protokollen können beliebige Objekte ausgetauscht werden. Da der externe Vermittler jedoch nur im Fehlerfall benötigt wird, ist es ein optimistisches Protokoll. Das neue Protokoll besitzt also Vorteile beider Arten von Austauschprotokollen.

Protokoll P6

Voraussetzungen: K besitzt Chipkarte C.

Implementierung: I1, I2-Hw, I3-Hw, I4-Hw, I5-Hw.

Eigenschaften: Optimistisches Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für K, T1/T0-Terminierung für H.

Im folgenden gebe ich zuerst die Implementierungen I2-Hw bis I5-Hw an und weise anschließend die vom Protokoll P6 erfüllten Eigenschaften nach. Danach diskutiere ich die Besonderheiten des neuen Protokolls.

5.2.1 Protokollbeschreibung

Für das Aushandeln des Austausches kann wieder auf die Implementierung I1 in Abschnitt 3.1.1 zurückgegriffen werden. Statt A und B heißen die beteiligten Parteien jetzt Kunde K und Händler H.

Implementierung I2-Hw: Das Vorbereiten des Austausches (siehe Tabelle 5.1) beginnt mit dem Transfer von O_H, O_K und der Beschreibung dieser Objekte an die Chipkarte C. Die Reihenfolge, in der diese Nachrichten gesendet werden, spielt dabei keine Rolle. Nach dem Empfang dieser Nachrichten überprüft die Chipkarte, ob beide Objekte ihrer Beschreibung entsprechen und unterbricht den Austausch, falls ein Fehler entdeckt wird. Der Kunde kann während I2-Hw jederzeit einen Abbruch mit I5-Hw erzwingen, ohne daß er auf die Nachricht von H warten muß.

Implementierung I3-Hw: Nach erfolgreicher Überprüfung in I2-Hw schickt die Chipkarte in I3-Hw (siehe Tabelle 5.2) O_K an den Händler, der ebenfalls die Gültigkeit dieses Objekts überprüft. Wenn ein Fehler entdeckt wird oder gar keine Nachricht ankommt, kann der Händler hier einfach den Austausch beenden. Ansonsten informiert der Händler die Chipkarte, daß er O_K erhalten hat. Die Chipkarte prüft den Nachweis des Empfangs und muß eine Konfliktlösung mit I4-Hw starten, falls keine oder eine fehlerhafte Nachricht empfangen wurde. Wenn kein Fehler aufgetreten ist, liefert die Chipkarte O_H an den Kunden aus.

Implementierung I4-Hw: Der Kunde kann die Konfliktlösung mit I4-Hw starten, wenn keine oder eine fehlerhafte Nachricht statt einer Empfangsbestätigung des Händlers eintrifft. Bei einem solchen Fehler wendet sich der Kunde mit seinem Wunsch nach Konfliktlösung an die Chipkarte, die sich gemäß dem Protokoll in Tabelle 5.3 verhält. Falls die Chipkarte I2-Hw noch nicht beendet hat, kann der Austausch nicht fortgesetzt werden und die Chipkarte beendet stattdessen den Austausch.

Andernfalls besitzt die Chipkarte ein gültiges O_K und O_H, so daß sie eine Konfliktlösung mit dem externer Vermittler startet. Nachdem der Vermittler O_K empfangen hat, braucht er dessen Gültigkeit nicht zu überprüfen, da die Chipkarte dies bereits erledigt hat. Stattdessen speichert der Vermittler O_K und versucht es an den Händler weiterzuleiten. Dann sendet der Vermittler eine Empfangsbestätigung an die Chipkarte, die daraufhin O_H an den Kunden ausliefert.

Implementierung I5-Hw: Der Kunde kann mit I5-Hw (siehe Tabelle 5.4) den Abbruch des Austausches veranlassen, solange I2-Hw noch nicht beendet ist. Dann kann die Chipkarte ohne die Beteiligung des externen Vermittlers den Austausch sofort abbrechen.

5.2.2 Nachweis der Protokolleigenschaften

In diesem Abschnitt werden die Eigenschaften des vorgestellten Protokolls gemäß der in Abschnitt 2.2.3 eingeführten Fairneßdefinition diskutiert.

I2-Hw: Implementierung von Modul M2 für Austausch mit sicherer Hardware		
H $\rightarrow$ C	:	O_H und Beschreibung von O_K
K $\rightarrow$ C	:	O_K und Beschreibung von O_H
C	:	Prüfe, ob O_H und O_K den Beschreibungen entsprechen (falls nein, beende Austausch)

Tabelle 5.1: Modul M2 für optimistisch fairen Austausch mit Chipkarte.

I3-Hw: Implementierung von Modul M3 für Austausch mit sicherer Hardware		
C $\rightarrow$ H	:	O_K
H	:	Prüfe, ob O_K der Beschreibung entspricht (falls nein, beende Austausch)
H $\rightarrow$ C	:	Nachweis des Empfangs NE(H, O_K)
C	:	Prüfe, ob NE(H, O_K) gültig ist (falls nein, starte I4-Hw)
C $\rightarrow$ K	:	O_H

Tabelle 5.2: Modul M3 für optimistisch fairen Austausch mit Chipkarte.

I4-Hw: Implementierung von Modul M4 für Austausch mit sicherer Hardware		
K $\rightarrow$ C	:	Austausch fortsetzen
C	:	Falls bereits abgebrochen oder I2-Hw noch nicht beendet:
		C $\rightarrow$ K : Austausch abgebrochen
		Sonst:
		C $\rightarrow$ V : O_K und Beschreibung von O_K
		V : Speichere O_K, liefere O_K an den Händler H
		V $\rightarrow$ C : Nachweis des Empfangs NE(V, O_K)
		C $\rightarrow$ K : O_H

Tabelle 5.3: Der Kunde versucht in dieser Implementierung von Modul M4 den Austausch fortzusetzen.

I5-Hw: Implementierung von Modul M5 für Austausch mit sicherer Hardware		
K $\rightarrow$ C	:	Austausch abbrechen
C	:	Falls I3-Hw bereits gestartet wurde:
		C $\rightarrow$ K : Fehler: starte I4-Hw
		Sonst:
		C $\rightarrow$ K : Austausch erfolgreich abgebrochen

Tabelle 5.4: Der Kunde kann mit dieser Implementierung von Modul M5 den Austausch abbrechen.

Wirksamkeit: Die Wirksamkeit ist offensichtlich erfüllt, da Kunde und Händler im fehlerfreien Fall die gewünschten Objekte in I3-Hw erhalten, die aufgrund der vorherigen Prüfung durch die Chipkarte auch garantiert ihrer Beschreibung entsprechen.

Terminierung: Die Terminierung des Protokolls kann nur vom Kunden erzwungen werden. Mit den Konfliktlösungsmodulen I4-Hw und I5-Hw kann der Kunde immer eine Beendigung der Protokollausführung erreichen, was T2-Terminierung sicherstellt.
Der Händler weiß dagegen nicht, ob der Kunde den Austausch weiterführen wird oder bereits abgebrochen hat, solange er nicht in I3-Hw oder I4-Hw das Objekt O_K erhält. Daher kann er, wenn er keine Antwort vom Kunden empfängt, den Austausch als abgebrochen ansehen und terminieren. Falls er sich geirrt hat, wird er später O_K vom Vermittler geliefert bekommen. In diesem Fall handelt es sich um T1-Terminierung.
Falls der Vermittler in I4-Hw nicht in der Läge ist, O_K dem Händler zuzustellen (z.B. weil er O_K nicht an einen terminierten Händler liefern kann), dann müßte der Händler den Vermittler bei einem nicht erfolgreichen Austausch regelmäßig fragen, ob der Kunde inzwischen eine Konfliktlösung durchgeführt hat. In diesem Fall kann der Händler nicht terminieren, so daß das Protokoll nur die Eigenschaft T0-Terminierung für den Händler besitzt.

Fairneß: Wenn der Kunde bei der Terminierung O_H besitzt, wird sich das nicht mehr ändern, so daß dieser Fall fair für ihn ist. Wenn der Kunde ohne O_H terminiert, dann hat er vor dem Start von I3-Hw erfolgreich I5-Hw aufgerufen, so daß der Händler O_K nicht mehr erhalten kann. Daher ist das Protokoll fair für den Kunden.
Wenn der Händler O_K vor der Terminierung erhält, wird sich das nicht mehr ändern, so daß dieser Fall fair für ihn ist. Wenn der Händler ohne O_K terminiert, dann hat er auch keinen Nachweis des Empfangs an den Kunden geschickt. Daher kann der Kunde O_H nur mit Hilfe des Vermittlers bekommen, der jedoch vorher O_K an den Händler liefern wird. Daher ist das Protokoll auch für den Händler fair.
Falls die Protokollvariante mit T0-Terminierung für den Händler ausgeführt wird, ist das Protokoll ebenfalls fair, da der Händler nach der Konfliktlösung des Kunden mit I4-Hw das Objekt O_K beim Vermittler abrufen kann, so daß nach der letzten Zustandsänderung immer ein fairer Zustand erreicht wird.
Insgesamt sichert das Protokoll F4-Fairneß für beide Parteien, da die Konfliktlösung mit Hilfe des Vermittlers automatisiert durchgeführt wird.

Nichtabstreitbarkeit: Das Protokoll bietet nur Nichtabstreitbarkeit des Empfangs für den Kunden, falls die Chipkarte $\mathrm{NE}(H, O_K)$ oder $\mathrm{NE}(V, O_K)$ an diesen ausliefert. Ansonsten ist die Nichtabstreitbarkeit in der beschriebenen Version des Austauschprotokolls nicht berücksichtigt. Die Ergänzungen für Nichtabstreitbarkeit der Herkunft aus Abschnitt 3.2.1 können aber auch hier angewendet werden. Ebenso könnte eine Erweiterung für Nichtabstreitbarkeit des Empfangs für den Händler gemäß den in Abschnitt 3.2.2 vorgestellten Ideen realisiert werden.

5.2.3 Diskussion

Gegenüber den in Abschnitt 3.1 vorgestellten Protokollen besitzt das hier vorgestellte Protokoll P6 die Vorteile, daß es sowohl optimistisch ist, als auch keine besonderen Anforderungen an die auszutauschenden Objekte stellt. Aufgrund seiner Eigenschaften eignet sich P6 besonders gut für den Austausch von elektronischem Geld gegen eine digitale Ware. Das Objekt des Kunden ist dann das Geld und der Händler liefert die Ware. Aus der Sicht des Händlers spricht auch nichts gegen die schwächere Form der Terminierung. Da der Kunde den Austausch zu einem beliebigen späteren Zeitpunkt fortsetzen kann, bleibt für den Händler die Möglichkeit erhalten, mit diesem Austauschvorgang Geld zu verdienen. Die Weiterleitung des Geldes in I4-Hw geschieht dann einfach durch das Einzahlen auf das Konto des Händlers.

Auch anonyme Zahlungen werden gut unterstützt, da der Kunde die Konfliktlösung immer ohne Deanonymisierung durchführen kann. Die Chipkarte hat die Ware bereits in I2-Hw geprüft, so daß eine Rückabwicklung (wie z.B. in P3–P5) nicht nötig ist. Falls der Austausch in I2-Hw scheitert, kann der Kunde sein immer noch anonymes Geld bei einem anderen Einkauf verwenden, da es nie die Chipkarte verlassen hat.

Andere optionale Anforderungen an die Benachteiligungsfreiheit erfüllt das Protokoll jedoch nicht. Bei verderblichen Waren kann eine Unterbrechung der Kommunikation zu einem Wertverlust der Ware führen. Dieses Problem wird erst durch die in Abschnitt 5.4 beschriebenen Protokolle gelöst.

5.3 Optimierung der Protokollausführung für Chipkarten

Im Vergleich zu einem PC oder anderen leistungsfähigen Computersystemen verfügt eine Chipkarte nur über wenig Speicher und eine geringe Rechenleistung. Daher ist die Ausführung des im vorherigen Abschnitt vorgestellten Protokolls nicht möglich, sobald die ausgetauschten Objekte die Grenzen des vorhandenen Speichers oder der Rechenleistung übersteigen. Um die Vorteile einer Chipkarte auch in einem solchen Fall nutzen zu können, werden in diesem Abschnitt Konzepte vorgestellt, wie sich diese Probleme unabhängig vom konkreten Austauschprotokoll lösen lassen.

Die Grundidee für diese Optimierungen besteht im Auslagern von schwierigen Aufgaben. Dabei übernimmt der Kunde oder der externe Vermittler soweit möglich Aufgaben der Chipkarte und entlastet sie dadurch.

5.3.1 Externe Speicherung der Objekte

Der nichtflüchtige Speicher einer Chipkarte hat heute normalerweise die Größe von wenigen Kilobytes und kann daher größere Objekte nicht aufnehmen. Falls zur Bearbeitung der Objekte der relativ kleine Arbeitsspeicher ausreicht und nur geringe Rechenleistung benötigt wird, kann die vollständige Speicherung der Objekte auch außerhalb der Chipkarte erfolgen, z.B. auf dem PC des Kunden, an den die Chipkarte angeschlossen ist. Die Idee besteht darin, daß die Chipkarte die empfangenen Objekte symmetrisch verschlüsselt an den Computer des Kunden schickt, wo sie gespeichert werden. Später kann

die Chipkarte ein verschlüsseltes Objekt wieder anfordern und entschlüsseln, sobald es benötigt wird. Die Verschlüsselung durch die Chipkarte stellt dabei sicher, daß keine Informationen vorzeitig bekannt werden.

Bei diesem Vorgehen muß die Chipkarte lediglich den zufällig gewählten symmetrischen Schlüssel speichern. Zusätzlich sollte sie auch noch den Hashwert des ausgelagerten Objekts berechnen und abspeichern, so daß beim erneuten Einlesen die Integrität des zurückgelieferten Objekts überprüft werden kann. Da ein zweites Objekt mit gleichem Hashwert praktisch nicht berechenbar ist, ist der Kunde gezwungen, wieder dasselbe Objekt an die Chipkarte zurückzuliefern.

Falls das Objekt später nur in kleinen Blöcken von der Chipkarte verarbeitet wird, sollten Hashwerte für die einzelnen Blöcke berechnet werden. Um Speicherplatz zu sparen können diese Hashwerte auch außerhalb der Chipkarte gespeichert werden, falls die Hashfunktion auf die unverschlüsselten Blöcke des Objekts angewendet wird. Der Kunde kann die ausgelagerten Blöcke dann nicht verändern, da er ohne Kenntnis des Schlüssels keinen Hashwert eines Klartexts berechnen kann. Um sich gegen das Vertauschen von Blöcken zu schützen, sollte die Chipkarte zusätzlich zum Klartext eines Blocks die Position dieses Blocks im Bezug auf das ganze Objekt mithashen. Dann ist die Integrität der ausgelagerten Daten stets überprüfbar und die Chipkarte muß lediglich den Dechiffrierschlüssel speichern.

Diese bisher diskutierten Maßnahmen stellen eine deutliche Reduzierung des Speicherbedarfs für die Chipkarte dar. Es muß jedoch darauf geachtet werden, daß kein neuer Engpaß entsteht: Da zum Beispiel die Datenübertragung von und zur Chipkarte gemäß ISO 7816 nur über eine Datenleitung erfolgt, die gewöhnlich nicht sehr schnell getaktet ist, kann das Auslagern eines Objekts mehrere langsame Datenübertragungen verursachen. Falls zum Beispiel in Schritt I2-Hw (siehe Tabelle 5.1) der Händler O_H schickt, wäre es ineffizient, wenn die Chipkarte dieses Objekt zuerst einliest und dann wieder auslagert. Stattdessen sollte ein solcher Protokollschritt, der keine direkte Verarbeitung des Objekts beinhaltet, wie folgt implementiert werden: Der Absender des Objekts – in dem Beispiel also der Händler – verschlüsselt sein Objekt und schickt es an den Computer des Kunden, wo es gespeichert wird. Der entsprechende Dechiffrierschlüssel wird dann vom Händler zur Chipkarte geschickt, wobei wegen der in Abschnitt 5.1.2 geforderten Verschlüsselung von Nachrichten die Vertraulichkeit des Schlüssels gewährleistet ist.

Im Unterschied zur bisherigen Lösung kann die Chipkarte den Hashwert des Objekts nicht mehr selbst berechnen. Dies kann dadurch kompensiert werden, daß sowohl der Kunde als auch der Händler jeweils den Hashwert des verschlüsselten Objekts berechnen und an die Chipkarte schicken. Diese speichert und vergleicht die beiden Hashwerte und akzeptiert nur dann, wenn beide identisch sind. Damit ist sichergestellt, daß Kunde und Händler dasselbe verschlüsselte Objekt besitzen, und die Chipkarte kann später überprüfen, ob sie genau dieses ausgelagerte Objekt vom Kunden erhält.

5.3.2 Delegieren der Überprüfung der Objekte

Andere Einschränkungen von Chipkarten sind ihre begrenzte Rechenleistung und der geringe Arbeitsspeicher, die aufwendige Berechnungen innerhalb die Chipkarte verhindern.

Daher sollte man den Umfang der Berechnungen, die die Chipkarte selbst ausführen muß, minimieren.

Durch das Delegieren der Überprüfung der Objekte kann die Chipkarte nicht mehr sofort ein fehlerhaftes Objekt erkennen. Stattdessen arbeitet die Chipkarte weiter, als ob die Überprüfung erfolgreich verlaufen wäre. Erst wenn sich eine Partei meldet und über ein Problem bei der Überprüfung berichtet, muß eine Fehlerbehandlung durchgeführt werden, die aus zwei Schritten besteht: Zuerst muß verifiziert werden, daß wirklich ein Fehler vorliegt, und dann muß die Fairneß sichergestellt werden.

Fehlermeldungen verifizieren

Ob wirklich ein Problem mit einem ausgetauschten Objekt vorliegt oder dies fälschlicherweise behauptet wird, kann nur eine vertrauenswürdige Partei prüfen, was zwei Möglichkeiten eröffnet:

1. Eine Möglichkeit besteht darin, daß die Chipkarte eine Überprüfung selbst durchführt, falls sie dazu in der Lage ist. In diesem Fall könnte auch eine längere Rechenzeit für die Überprüfung akzeptabel sein, solange diese Art der Fehlerbehandlung nicht häufiger in Anspruch genommen wird.

2. Die Alternative besteht im Einschalten eines externen Vermittlers, der die Überprüfung übernimmt. Diese Lösung muß immer dann genutzt werden, wenn die Chipkarte allein keine Überprüfung durchführen kann. Der deutlich leistungsfähigere externe Vermittler teilt schließlich der Chipkarte das Ergebnis der Überprüfung mit, die damit über die Fortsetzung des Austauschprotokolls entscheiden kann.

Im Unterschied zu den Leistungen, die ein externer Vermittler bei der Unterstützung des fairen Austausches erbringt, handelt es sich bei der Überprüfung der Objekteigenschaften um eine leichter zu implementierende Dienstleistung: Die einzige Anforderung an den Vermittler besteht darin, daß seine Berechnungen korrekt sind. Der Vermittler muß das Ergebnis jedoch nicht speichern, sondern nur der Chipkarte mitteilen. Daher können sogar O_K und O_H von verschiedenen Vermittlern unabhängig voneinander überprüft werden. Dies ermöglicht eine gute Lastverteilung durch die Bereitstellung dieser Dienstleistung auf verschiedenen unabhängigen Servern.

Zum Schluß noch eine Bemerkung zum Nutzen dieser Maßnahme: Für die effiziente Protokollausführung ist es gerade bei einer Auslagerung der Überprüfung an einen externen Vermittler wichtig, daß sie nur sehr selten in Anspruch genommen wird. Ähnlich wie bei optimistischen Protokollen mutiert das Austauschprotokoll sonst zu einem aktiven Protokoll. Dann wäre aber der direkte Einsatz eines Protokolls mit aktivem Vermittler die bessere Lösung gewesen.

Fairneß sicherstellen

Falls eine Partei ein Objekt ablehnt und die anschließende Überprüfung seine Gültigkeit nachweist, so müssen dieselben Konfliktlösungsmechanismen gestartet werden wie beim

Abbruch der Kommunikation direkt nach dem Senden des Objekts. Es macht nämlich keinen Unterschied, ob eine Partei eine falsche Antwort zurückliefert oder gar keine.

Falls bei der Überprüfung eines Objekts nachweislich ein Fehler entdeckt wurde, muß eine Fehlerbehandlung die Fairneß nachträglich sicherstellen: Wurde bereits vorher ein gültiges Objekt ausgeliefert, so muß dies nun widerrufen werden. Erhält dagegen das erste ausgelieferte Objekt einen Fehler, muß die Chipkarte lediglich die Auslieferung des zweiten Objekts unterbinden.

Beispiel

Im Austauschprotokoll P6 prüfen sowohl die Chipkarte in I2-Hw als auch direkt anschließend der Händler in I3-Hw die Gültigkeit von O_K. Daher kann die Chipkarte auf die Überprüfung verzichten und einfach auf die Antwort des Händlers warten. Falls dieser einen Fehler beim gelieferten O_K meldet, wird die Chipkarte dies wie oben beschrieben verifizieren. Bestätigt sich dieser Fehler, so wird das Austauschprotokoll beendet, ohne O_H auszuliefern. Andernfalls kann die Chipkarte mit dem Konfliktlösungsprotokoll I4-Hw die Fairneß beim Austausch sicherstellen. Dazu muß dann allerdings auch der Vermittler in I4-Hw eine Überprüfung des empfangenen O_K durchführen, weil er sich nicht auf die vorherige Überprüfung durch die Chipkarte verlassen kann.

5.4 Austausch verderblicher Ware

Alle bisher vorgestellten Protokolle haben die Einschränkung, daß der Wert der ausgetauschten Objekte als zu jeder Zeit konstant angenommen wird. Verderbliche Objekte besitzen diese Eigenschaft jedoch nicht. In diesem Abschnitt zeige ich, wie der Austausch von verderblichen Objekten mit der Unterstützung von sicherer Hardware realisiert werden kann.

Zuerst diskutiere ich die besondere Problematik von verderblichen Objekten, um dann in Abschnitt 5.4.2 den Lösungsansatz mit sicherer Hardware vorzustellen. Das in Abschnitt 5.4.3 präsentierte Protokoll erlaubt speziell den Austausch von verderblichen Waren gegen digitales Geld. Schließlich wird in Abschnitt 5.4.4 noch eine Protokollvariante für besonders schnell verderbliche Waren vorgeschlagen.

5.4.1 Das Problem verderblicher Objekte

Der Wert eines Objekts ist häufig abhängig vom Zeitpunkt an dem man es erhält. Staumeldungen verlieren zum Beispiel ihren Wert sobald sich der Stau aufgelöst hat, und ein Flugticket wird für den Käufer wertlos, wenn er es erst nach dem Start des Flugzeugs erhält. Ein anderes Beispiel für zeitkritischen Objekte sind aktuelle Börseninformationen, die bei verzögerter Auslieferung ihren Wert für den Käufer verlieren, weil sich diese Informationen bereits nach kurzer Zeit in den Börsenkursen der betroffenen Unternehmen widerspiegeln. Objekte mit einem solchen Wertverlust im Laufe der Zeit nenne ich *verderbliche Objekte*.

Das Problem mit diesen verderblichen Objekten besteht darin, daß normalerweise niemand außer dem Empfänger weiß, wann das Objekt angekommen ist. Der Empfänger kann immer behaupten, daß es eine Zeitverzögerung bei der Übertragung gab und das Objekt erst ankam, als es schon wertlos war. Es gibt jedoch keine Möglichkeit jemanden davon zu überzeugen, daß so eine Verspätung eingetreten ist. Ebenso könnte man auch fälschlicherweise behaupten, man hätte das Objekt zu spät erhalten. Auch ein aktiv am Austausch beteiligter Vermittler kann nur die Zeit dokumentieren, zu der er ein Objekt abgeschickt hat. Ob es den Empfänger erreicht hat und wann dies passierte, kann er jedoch auch nicht mit ausreichender Sicherheit feststellen. Aus diesen Gründen hat schon Asokan festgestellt [Aso98, Seite 33], daß fairer Austausch von verderblichen Objekten ein schwieriges Problem darstellt, das bisher ungelöst ist. In diesem Abschnitt entwickle ich Lösungen für dieses Problem, die auf dem Einsatz von sicherer Hardware basieren.

5.4.2 Der Lösungsansatz

Das Problem mit verderblichen Objekten besteht darin, daß sich sehr schwer abschätzen läßt, welchen Wert ein Objekt zu einem bestimmten Zeitpunkt besitzt. Außerdem werden verschiedene Personen den Wert eines Objekts nicht in gleicher Weise beurteilen. Aus diesen Gründen sollte die Entscheidung über den Wert eines Objekts dem Empfänger überlassen werden. Am zuverlässigsten kann diese Entscheidung über den Wert gefällt werden, sobald das Objekt beim Empfänger angekommen ist. Deshalb sollte es sich bei dem Empfänger um den Käufer handeln, der im Besitz einer Chipkarte ist. Die Chipkarte kann dann vor der Auslieferung des verderblichen Objekts an den Käufer nachfragen, ob dieser das verderbliche Objekt immer noch zu den vorher festgelegten Konditionen haben will. Wenn das der Fall ist, liefert die Chipkarte das verderbliche Objekt sofort. Andernfalls muß die Chipkarte den Austausch rückgängig machen.

Der Vorteil dieser Lösung, die Entscheidung lokal zwischen Chipkarte und dem Kunden zu treffen, liegt darin, daß keine exakte Zeitmessung notwendig ist. Sobald eine andere Partei, die nur über ein unzuverlässiges Kommunikationsmedium zu erreichen ist, an der Entscheidung über den Wert des Objekts beteiligt werden soll, müßte diese Partei den exakten Zeitpunkt des Empfangs des Objekts mitgeteilt bekommen. Eine sichere Hardware, die diesen Zeitpunkt exakt und manipulationssicher mitprotokollieren kann, läßt sich jedoch nur mit deutlich größeren Kosten realisieren, weswegen solche aufwendigeren Lösungsansätze hier nicht diskutiert werden. Alle im folgenden vorgestellten Protokolle kommen ohne Änderungen an der in Abschnitt 5.1.2 beschriebenen Hardware aus.

5.4.3 Ein Protokoll zum Austausch verderblicher Ware

Im diesem Abschnitt wird ein Protokoll vorgestellt, mit dem der Kunde sein Geld gegen eine verderbliche Ware des Händlers austauschen kann. Bei diesem Protokoll P7 handelt es sich um eine Variante von P6, die zwingend starke Widerrufbarkeit (siehe Abschnitt 2.3.4) des verwendeten Geldes voraussetzt, damit eine spätere Konfliktlösung möglich ist. Ansonsten müssen im Vergleich zu P6 keine zusätzlichen Voraussetzungen bezüglich Geld oder Ware erfüllt sein.

Protokoll P7

Voraussetzungen: K besitzt Chipkarte C, das Geld des Kunden K ist stark widerrufbar.

Implementierung: I1, I2-Hw, I3-Hw-verderb, I4-Hw-verderb, I5-Hw-verderb.

Eigenschaften: Optimistisches Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für K, T1-Terminierung für H.

Im folgenden gebe ich zuerst die gegenüber P6 geänderten Implementierungen an und diskutiere die vom Protokoll P7 erfüllten Eigenschaften.

Protokollbeschreibung

Das Protokoll für den Austausch verderblicher Ware ist eine Erweiterung von P6 und verwendet daher ebenfalls die Modulimplementierungen I1 und I2-Hw. Bei O_K handelt es sich jedoch um das Geld und bei O_H um die Ware.

Implementierung I3-Hw-verderb: Diese Implementierung erweitert I3-Hw im wesentlichen um eine Rückfrage, ob der Kunde die verderbliche Ware noch haben will. Die Chipkarte schickt in I3-Hw-verderb (siehe Tabelle 5.5) zuerst das Geld an den Händler, der eine Überprüfung durchführt. Wenn das Geld gültig ist, bestätigt der Händler den Empfang. Bevor die Chipkarte dann die Ware an den Kunden ausliefert, fragt sie zuerst den Kunden, ob er die Ware immer noch kaufen will oder ob sie inzwischen wertlos geworden ist. Falls der Kunde die Ware haben will, erhält er sie sofort von der Chipkarte. Die Zeitverzögerung zwischen der Bestätigung der Kaufentscheidung durch den Kunden und der unmittelbar danach folgenden Auslieferung der Ware ist offensichtlich minimal. Dabei wird lediglich lokale Kommunikation zwischen dem Kunden und der Chipkarte verwendet, so daß eine Unterbrechung durch einen Angreifer oder einen Übertragungsfehler ausgeschlossen werden kann. Daher kann der Kunde sicher sein, daß die ausgelieferte Ware immer noch wertvoll ist.

Implementierung I4-Hw-verderb: Der Kunde kann die Konfliktlösung mit I4-Hw-verderb starten, wenn keine oder eine fehlerhafte Nachricht statt einer Empfangsbestätigung des Händlers eintrifft. Bei einem solchen Fehler wendet sich der Kunde mit seinem Wunsch nach Konfliktlösung durch Fortsetzen des Austausches an die Chipkarte, die sich gemäß dem Protokoll in Tabelle 5.6 verhält. Falls die Chipkarte I2-Hw noch nicht beendet hat, kann der Austausch nicht fortgesetzt werden und die Chipkarte beendet stattdessen den Austausch.

Andernfalls besitzt die Chipkarte Geld und Ware, die den Beschreibungen entsprechen, und startet eine Konfliktlösung mit dem externer Vermittler. Nachdem der Vermittler das Geld empfangen hat, zahlt er es auf das Konto des Händlers ein, so daß dieser immer automatisch das Geld erhält. Dann sendet der Vermittler eine Empfangsbestätigung für das Geld an die Chipkarte, die vor der Auslieferung der Ware erst nachfragt, ob dieser

I3-Hw-verderb: Implementierung von Modul M3 für verderbliche Ware		
C → H	:	Geld
H	:	Prüfe, ob das Geld der Beschreibung entspricht (falls nein, beende Austausch)
H → C	:	Nachweis des Empfangs NE(H,Geld)
C	:	Prüfe, ob NE(H,Geld) gültig ist (falls nein, starte I4-Hw-verderb oder I5-Hw-verderb)
C → K	:	Soll die Ware ausgeliefert werden?
K	:	Entscheide über Wert der Ware (falls wertlos, starte I5-Hw-verderb)
K → C	:	Ja, Ware sofort ausliefern!
C → K	:	Ware

Tabelle 5.5: Modul M3 für optimistisch fairen Austausch einer verderblichen Ware.

I4-Hw-verderb: Implementierung von Modul M4 für verderbliche Ware				
K → C	:	Austausch fortsetzen		
C	:	Falls bereits abgebrochen oder I2-Hw noch nicht beendet:		
		C → K	:	Austausch abgebrochen
		Sonst:		
		C → V	:	Geld und dessen Beschreibung
		V	:	Geld auf Händlerkonto einzahlen
		V → C	:	Nachweis des Empfangs NE(V,Geld)
		C → K	:	Soll die Ware ausgeliefert werden?
		K	:	Entscheide über Wert der Ware (falls wertlos, starte I5-Hw-verderb)
		K → C	:	Ja, Ware sofort ausliefern!
		C → K	:	Ware

Tabelle 5.6: Auch beim Austausch einer verderblichen Ware kann der Kunde das Fortsetzen des Austausches mit Modul M4 versuchen.

I5-Hw-verderb: Implementierung von Modul M5 für verderbliche Ware				
K → C	:	Austausch abbrechen		
C	:	Falls die Ware bereits an den Kunden geliefert wurde:		
		C → K	:	Austausch bereits beendet
		Falls I2-Hw noch nicht beendet wurde:		
		C → K	:	Austausch abgebrochen
		Sonst:		
		C → V	:	Geld und dessen Beschreibung
		V	:	Widerrufe das Geld
		V → C	:	Bestätigung des Widerrufs
		C	:	Lösche Ware, breche Austausch ab
		C → K	:	Austausch abgebrochen, Bestätigung des Widerrufs

Tabelle 5.7: Falls eine verderbliche Ware ihren Wert verloren hat, kann der Kunde mit Modul M5 der Austausch zurücksetzen.

die Ware noch für wertvoll erachtet. Wenn der Kunde die Ware ablehnt, muß der Austausch mit I5-Hw-verderb zurückgesetzt werden. Ansonsten bekommt der Kunde sofort die gewünschte Ware.

Implementierung I5-Hw-verderb: Der Kunde kann mit I5-Hw-verderb (siehe Tabelle 5.7) den Abbruch des Austausches veranlassen. Falls die Ware bereits ausgeliefert wurde, ist es für ein Zurücksetzen des Austausches zu spät. Falls I2-Hw noch nicht beendet ist, kann die Chipkarte sofort den Austausch abbrechen. In allen anderen Fällen wird die Beteiligung des externen Vermittlers benötigt, der das Geld des Kunden widerrufen muß. Die Chipkarte sendet dazu das Geld an den Vermittler, der dann den Widerruf des Geldes bei der Bank veranlaßt, die das verwendete elektronische Zahlungssystem betreibt. Der Vermittler informiert die Chipkarte über den erfolgten Widerruf und diese löscht daraufhin die gespeicherte Ware. Schließlich unterrichtet die Chipkarte den Kunden über den Erfolg des Widerrufs und übergibt ihm gegebenenfalls noch die Informationen, die dem Kunden ein erneutes Ausgeben des Geldes erlauben.

Diskussion der Protokolleigenschaften

Bevor die Fairneßeigenschaften im einzelnen untersucht werden, diskutiere ich zuerst die Eignung des Protokolls für verderbliche Ware. In Protokoll P7 wird die Ware zuerst in I2-Hw von der Chipkarte überprüft, ob sie der geforderten Beschreibung entspricht. Das allein genügt jedoch nicht bei einer verderblichen Ware, da zwischen der Überprüfung in I2-Hw und der Auslieferung in I3-Hw-verderb bzw. I4-Hw-verderb viel Zeit vergehen kann. Durch die Kommunikation mit dem Händler bzw. dem Vermittler kann es zu Verzögerungen kommen, die der Kunde nicht akzeptiert. Daher kann er immer direkt vor der Auslieferung der Ware seine endgültige Entscheidung treffen, ob er die Ware noch haben will oder nicht. Bei einer positiven Entscheidung erhält der Kunde die Ware sofort ausgeliefert. Der wesentliche Unterschied zu anderen Protokollen besteht darin, daß zwischen der Entscheidung des Kunden und der Auslieferung der Ware keine Kommunikation mit einer externen Partei nötig ist. Deshalb können nur noch vom Kunden selbst verursachte Verzögerungen eintreten, was dann jedoch dessen eigene Schuld ist.

Falls der Kunde den Wert einer Ware für zu stark gesunken erachtet, kann er mit I5-Hw-verderb den Austausch immer noch Abbrechen, solange er die Ware nicht erhalten hat. Auf diese Weise wird jede Benachteiligung des Kunden vermieden.

Das hier vorgestellte Protokoll ist fair für den Kunden, da er immer sein Geld zurückbekommen kann, wenn er die Ware nicht erhält oder ihren Empfang ablehnt. Ebenso ist die Fairneß aus der Sicht des Händlers gewährleistet, da die Chipkarte nur genau dann das Geld widerruft, wenn sie die Ware noch nicht ausgeliefert hat.

Im folgenden werden noch kurz die einzelnen Fairneßeigenschaften des Protokolls für verderbliche Ware diskutiert:

Wirksamkeit: Die Wirksamkeit ist erfüllt, da Kunde und Händler im fehlerfreien Fall die gewünschten Objekte in I3-Hw-verderb erhalten, die aufgrund der vorherigen Prüfung durch die Chipkarte auch garantiert ihrer Beschreibung entsprechen.

Terminierung: Mit den Konfliktlösungsmodulen I4-Hw-verderb und I5-Hw-verderb kann der Kunde immer eine Beendigung der Protokollausführung erreichen, was T2-Terminierung sicherstellt.
Auch der Händler wird in jedem Fall terminieren. Wenn der Händler mit dem Geld des Kunden terminiert, besteht die Einschränkung, daß der Kunde das Geld noch widerrufen kann, wenn er die Ware mit I5-Hw-verderb zurückweist. Dann handelt es sich um Terminierung der Form T1–.
Wenn der Händler ohne das Geld des Kunden terminiert, weiß er dagegen nicht, ob der Kunde den Austausch weiterführen wird, solange er nicht in I4-Hw-verderb das Geld erhält. In diesem Fall handelt es sich also um Terminierung der Form T1+.

Fairneß: Wenn der Kunde bei der Terminierung die Ware besitzt, wird sich das nicht mehr ändern, so daß dieser Fall fair für ihn ist. Wenn der Kunde ohne die Ware terminiert, dann hat er erfolgreich I5-Hw-verderb aufgerufen, so daß der Händler das Geld entweder nie erhalten hat oder wegen eines Widerrufs nicht mehr besitzt. Daher ist das Protokoll fair für den Kunden.
Wenn der Händler das Geld bei der Terminierung besitzt, ist dieser Zustand für ihn fair. Wegen der Terminierung T1– kann das Geld jedoch noch vom Kunden mittels I5-Hw-verderb widerrufen werden. Da dies nur dann möglich ist, wenn der Kunde die Ware beweisbar nicht erhalten hat, ist auch dieser Fall für den Händler fair.
Wenn der Händler ohne das Geld terminiert, dann hat er in I3-Hw-verderb auch keinen Nachweis des Empfangs an den Kunden geschickt. Daher kann der Kunde die Ware nur in I4-Hw-verderb mit Hilfe des Vermittlers bekommen, der jedoch vorher das Geld auf das Händlerkonto einzahlt. Daher ist das Protokoll auch für den Händler fair.
Insgesamt sichert das Protokoll F4-Fairneß für beide Parteien, da die Konfliktlösung mit Hilfe des Vermittlers automatisiert durchgeführt wird.

Nichtabstreitbarkeit: Die Nichtabstreitbarkeit des Empfangs für den Kunden ist bei diesem Austauschprotokoll nur dann erfüllt, wenn die Chipkarte zusammen mit der Ware immer NE(H,Geld) bzw. NE(V,Geld) ausliefert. In diesem Fall muß auch kein späteres Widerrufen des Geldes befürchtet werden, so daß der Nachweis des Empfangs seine Aussagekraft behält. Ein Nachweis des Empfangs für den Händler macht hier keinen Sinn, da die Auslieferung der Ware lokal zwischen Chipkarte und Kunde erfolgt. Falls der Kunde die Ware für wertlos hält, kann er immer noch den Widerruf der Zahlung veranlassen, was auch den Nachweis des Empfangs für den Händler entwerten müßte.
Die Ergänzungen für Nichtabstreitbarkeit der Herkunft aus Abschnitt 3.2.1 können auch hier angewendet werden, was aber hauptsächlich bei der Ware sinnvoll ist.

Die sehr schwache Form der T1-Terminierung für den Händler sollte bei der praktischen Umsetzung des Protokolls durch ein Zeitlimit entschärft werden: Wie bereits in Abschnitt 4.2.3 erläutert, sollte dazu eine ausreichend große Zeitspanne gewählt werden, während der der Kunde immer erfolgreich eine Konfliktlösung mit dem Vermittler durchführen kann. Nach Ablauf der Zeitspanne wird sich dann der Zustand des Händlers nicht mehr

ändern. In der Praxis bedeutet selbst ein relativ kurz gewähltes Zeitlimit (z.B. 2 Wochen) keine wesentliche Einschränkung für den Kunden, da dieser beim Austausch einer verderblichen Ware sowieso an einer möglichst frühzeitigen Konfliktlösung interessiert ist.

5.4.4 Minimierung der Verzögerung beim Austausch von verderblichen Waren

Bei dem in vorherigen Abschnitt vorgestellten Austauschprotokoll für verderbliche Waren empfängt die Chipkarte im ersten Schritt von I2-Hw die Ware, während der Kunde erst am Ende von I3-Hw-verderb entscheiden kann, ob er die Ware immer noch haben will. Dazwischen muß erst noch das Geld an den Händler geschickt und auf dessen Antwort gewartet werden. Diese Kommunikation mit dem Händler verursacht möglicherweise eine Zeitverzögerung, die eine entscheidende Alterung der Ware herbeiführen kann. Bei schnell verderblichen Waren, die bereits nach kurzer Zeit ihren Wert verlieren (zum Beispiel Echtzeitbörsenkurse), sollte deshalb ein Austauschprotokoll verwendet werden, das die Zeitdifferenz zwischen der Lieferung der Ware an die Chipkarte und deren Auslieferung an den Kunden minimiert. Insbesondere die Kommunikation mit externen Parteien sollte vermieden werden, da dies zu unkalkulierbaren Verzögerungen führen kann.

In diesem Abschnitt wird ein Protokoll beschrieben, das diese Verzögerung minimiert und sich daher speziell für schnell verderbliche Waren eignet.

Protokoll P8

Voraussetzungen: K besitzt Chipkarte C, das Geld des Kunden K ist stark widerrufbar.

Implementierung: I1, I2-Hw-min, I3-Hw-min, I5-Hw-min.

Eigenschaften: Optimistisches Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für K, T1-Terminierung für H.

Im folgenden beschreibe ich zuerst die Implementierung von P8 und diskutiere dann die Eigenschaften des Protokolls.

Protokollbeschreibung

Das Protokoll für den Austausch verderblicher Ware orientiert sich einerseits am Protokoll P5 aus Abschnitt 3.1.6 und andererseits an den Ideen zum Delegieren der Objektüberprüfung aus Abschnitt 5.3.2. Zum Aushandeln verwendet es wie alle anderen Protokolle die Modulimplementierung I1.

Implementierung I2-Hw-min: Zum Vorbereiten des Austausches in I2-Hw-min (siehe Tabelle 5.8) muß lediglich die Chipkarte informiert werden, was ausgetauscht wird und welches Geld der Kunde liefert.

Implementierung I3-Hw-min: Die Chipkarte schickt in I3-Hw-min (siehe Tabelle 5.9) zuerst das Geld an den Händler, der eine Überprüfung durchführt. Wenn das Geld gültig ist, liefert der Händler im Gegenzug die Ware. Bevor die Chipkarte diese Ware an den Kunden ausliefert, fragt sie zuerst den Kunden, ob er die Ware immer noch kaufen will oder ob sie inzwischen wertlos geworden ist. Falls der Kunde die Ware haben will, erhält er sie sofort von der Chipkarte. Die Zeitverzögerung zwischen dem Empfang der Ware durch die Chipkarte und der Auslieferung der Ware ist offensichtlich minimal und nicht abhängig von der Kommunikation mit externen Parteien. Daher ist auch eine schnell verderbliche Ware mit hoher Wahrscheinlichkeit noch nicht veraltet.
Zum Schluß muß der Kunde die erhaltene Ware noch Überprüfen, da die Chipkarte diese Aufgabe an ihn delegiert hat. Falls jetzt ein Fehler entdeckt wird, muß der Austausch mit I5-Hw-min zurückgesetzt werden.

Implementierung I5-Hw-min: Der Kunde kann mit I5-Hw-min (siehe Tabelle 5.10) den Abbruch des Austausches veranlassen. Falls I3-Hw-min noch nicht gestartet wurde, kann die Chipkarte sofort den Austausch abbrechen.

Falls die Ware bereits ausgeliefert wurde, hat der Kunde vermutlich einen Fehler an der Ware entdeckt und will deshalb sein Geld zurück. In diesem Fall delegiert die Chipkarte die Überprüfung der Ware an den Vermittler. Falls dieser den Fehler nachvollziehen kann widerruft er das Geld. Andernfalls muß der Vermittler nichts unternehmen, da der Kunde eine korrekte Ware erhalten hat und der Austausch somit bereits beendet ist.

Falls die Ware noch nicht in I3-Hw-min ausgeliefert wurde, muß der Vermittler lediglich das Geld des Kunden widerrufen, um den Austausch zurückzusetzen.

Diskussion der Protokolleigenschaften

Das Protokoll P8 besitzt die folgenden Fairneßeigenschaften:

Wirksamkeit: Die Wirksamkeit ist erfüllt, da Kunde und Händler im fehlerfreien Fall die gewünschten Objekte in I3-Hw-min erhalten.

Terminierung: Mit dem Konfliktlösungsmodul I5-Hw-min kann der Kunde immer die Beendigung der Protokollausführung erreichen, was T2-Terminierung sicherstellt.
Der Händler terminiert, wenn er in I3-Hw-min kein Geld erhält oder nachdem er die Ware an den Kunden geschickt hat. Da in diesem zweiten Fall der Kunde das Geld noch widerrufen kann, wenn er die Ware mit I5-Hw-verderb zurückweist, handelt es sich um die Terminierung der Form T1– für den Händler.

Fairneß: Wenn der Kunde bei der Terminierung die Ware besitzt, wird sich das nicht mehr ändern, so daß dieser Fall fair für ihn ist. Wenn der Kunde ohne die Ware terminiert, dann hat er erfolgreich I5-Hw-min aufgerufen, so daß der Händler das Geld entweder nie erhalten hat oder wegen eines Widerrufs nicht mehr besitzt. Daher ist das Protokoll fair für den Kunden.
Wenn der Händler das Geld bei der Terminierung besitzt, ist dieser Zustand für ihn fair. Wegen der Terminierung T1– kann das Geld jedoch noch vom Kunden mittels

I2-Hw-min: Implementierung von Modul M2 für schnell verderbliche Ware		
K → C	:	Geld, Beschreibungen beider Objekte
C	:	Geld und Beschreibungen speichern

Tabelle 5.8: Modul M2 für optimistisch fairen Austausch einer schnell verderblichen Ware.

I3-Hw-min: Implementierung von Modul M3 für schnell verderbliche Ware		
C → H	:	Geld
H	:	Prüfe, ob das Geld der Beschreibung entspricht (falls nein, beende Austausch)
H → C	:	Ware
C → K	:	Soll die Ware ausgeliefert werden?
K	:	Entscheide über Wert der Ware (falls wertlos, starte I5-Hw-min)
K → C	:	Ja, Ware sofort ausliefern!
C → K	:	Ware
C	:	Prüfe, ob die Ware der Beschreibung entspricht (falls nein, starte I5-Hw-min)

Tabelle 5.9: Modul M3 für optimistisch fairen Austausch einer schnell verderblichen Ware.

I5-Hw-min: Implementierung von Modul M5 für schnell verderbliche Ware		
K → C	:	Austausch abbrechen
C	:	Falls I3-Hw-min noch nicht gestartet wurde:
		C → K : Austausch abgebrochen
		Falls die Ware bereits an den Kunden geliefert wurde:
		C → V : Ware, Geld und die Beschreibungen
		V : Überprüfe die Ware (falls Ware in Ordnung, Fehler: Austausch bereits beendet)
		Geld widerrufen (wegen fehlerhafter Ware)
		V → C : Bestätigung des Widerrufs
		C : Lösche Ware, breche Austausch ab
		C → K : Austausch abgebrochen, Bestätigung des Widerrufs
		Sonst:
		C → V : Geld und dessen Beschreibung
		V : Widerrufe das Geld
		V → C : Bestätigung des Widerrufs
		C : Lösche Ware, breche Austausch ab
		C → K : Austausch abgebrochen, Bestätigung des Widerrufs

Tabelle 5.10: Falls eine schnell verderbliche Ware ihren Wert verloren hat, kann der Kunde mit Modul M5 der Austausch zurücksetzen.

I5-Hw-min widerrufen werden. Da dies nur dann möglich ist, wenn der Kunde beweisbar keine der Beschreibung entsprechende Ware erhalten hat, ist auch dieser Fall für den Händler fair.
Wenn der Händler ohne das Geld terminiert, dann hat er in I3-Hw-min auch keine Ware an den Kunden geschickt, so daß das Protokoll auch für den Händler fair ist.
Insgesamt sichert das Protokoll F4-Fairneß für beide Parteien, da die Konfliktlösung mit Hilfe des Vermittlers automatisiert durchgeführt wird.

Nichtabstreitbarkeit: Die Nichtabstreitbarkeit ist in der beschriebenen Version des Austauschprotokolls nicht berücksichtigt. Für mögliche Ergänzungen für Nichtabstreitbarkeit der Herkunft und des Empfangs gilt das gleiche wie bei Protokoll P7 in Abschnitt 5.4.3.

Gegen die sehr schwache Form der Terminierung für den Händler hilft auch hier wieder genauso wie in Protokoll P7 ein Zeitlimit, bis zu dem der Kunde den Widerruf seines Geldes ausgeführt haben muß.

Diskussion

Dieses Protokoll hat die besondere Eigenschaft, daß es im fehlerfreien Fall mit der minimalen Anzahl von zwei Nachrichten zwischen Chipkarte und Händler in Modul M2 und M3 auskommt. Im Gegensatz zu Protokoll P5, das auch nur zwei Nachrichten benötigt, wird hier jedoch nur ein widerrufbares Objekt vorausgesetzt.

Das Protokoll P8 eignet sich besonders gut für wenig leistungsfähige Chipkarten bzw. sehr aufwendige Überprüfungen der Objekte, da das Protokoll die Objektüberprüfung vollständig an andere Parteien delegiert. Der Händler und der Vermittler führen die Überprüfung des Geldes aus, und Kunde und Vermittler sind für das Überprüfen der Ware zuständig. Wenn zusätzlich noch die Ideen zur externen Speicherung der Objekte zum Einsatz kommen, dann stellt das Protokoll nur minimale Anforderungen an die Ressourcen der Chipkarte.

5.5 Ein Protokoll mit T2-Terminierung für beide Parteien

Eine Vielzahl digitaler Waren, wie z.B. Bilder, Musik, Videos, Software, Nachrichten, etc., kann der Händler beliebig oft an verschiedene Kunden verkaufen. Es gibt bei diesen Waren keine Einschränkung, bis zu welchem Zeitpunkt der Kunde sich zum Kauf entschieden haben oder in welcher zeitlichen Reihenfolge der Händler sie verkaufen muß. Daher waren alle bisher in diesem Kapitel beschriebenen Protokolle so ausgelegt, daß nur der Kunde die Terminierung des Austausches durch ein Konfliktlösungsprotokoll erzwingen kann. Der Händler weiß also nie, ob nicht doch noch ein unvollständiger Austauschvorgang von einem Kunden zuendegeführt wird, was jedoch bei den oben genannten Waren keine wesentliche Einschränkung für den Händler darstellt.

Es gibt aber auch Waren, die nur in begrenzter Stückzahl vorhanden sind, wie zum Beispiel Kinokarten oder Flugtickets. Im Fall einer Sitzplatzreservierung existiert sogar

nur ein einziges Exemplar der Ware, das verkauft werden kann. Daher ist es für den Händler inakzeptabel, dem Kunden die Entscheidung zu überlassen, ob er etwas kaufen will oder nicht. Stattdessen muß der Händler die Terminierung des Austausches erzwingen können, wenn der Kunde den Austausch nicht innerhalb einer gewissen Zeitspanne beendet.

Eine einfache Zeitbeschränkung für die Gültigkeit des Austauschangebots des Händlers reicht oft nicht aus, da der Händler in diesem Fall bereits zu Beginn des Austausches festlegen muß, bis zu welchem Zeitpunkt der Kunde eine Konfliktlösung beim Vermittler veranlassen kann. Falls eine solche Zeitbeschränkung zu lang gewählt ist, bleibt der Händler möglicherweise bis zum Ablauf der Zeitbeschränkung über den Ausgang des Austausches im Unklaren. Falls die Zeitbeschränkung zu kurz ist, wird möglicherweise ein Abbruch erzwungen, obwohl der Kunde die Ware noch kaufen wollte. Ein weiteres Problem bei einem durch Zeitablauf erzwungenen Abbruch besteht darin, daß der Händler eventuell das Geld des Kunden schon eingezahlt hat. Deswegen müßte diese Zahlung widerrufen werden, falls der Kunde nachweisen kann, daß er die Ware nicht bekommen hat.

Wie auch der Händler die Terminierung eines Austausches erzwingen kann, wird im folgenden Protokoll P9 beschrieben. Sobald eine Partei nicht mehr länger auf Nachrichten von der anderen Partei warten will, kann sie sich an den Vermittler wenden, der den Austauschvorgang so schnell wie möglich zu Ende führt.

Protokoll P9

Voraussetzungen: K besitzt Chipkarte C.

Implementierung: I1, I2-Hw-term, I3-Hw-term, I4-Hw-term, I5-Hw-term.

Eigenschaften: Optimistisches Protokoll, F4-Fairneß für beide Parteien, T2-Terminierung für beide Parteien.

Im folgenden beschreibe ich zuerst die Implementierung von P9 und diskutiere dann die Eigenschaften des Protokolls. Da die ausgetauschten Objekte keine besonderen Eigenschaften wie Widerrufbarkeit besitzen müssen, verwende ich hier wieder die allgemeineren Bezeichnungen O_K und O_H für die Objekte von Kunde und Händler. Dadurch soll angedeutet werden, daß das Protokoll nicht nur für den Austausch Geld gegen Ware geeignet ist, sondern auch z.B. für eine elektronische Vertragsunterzeichnung mit einer Signaturchipkarte.

5.5.1 Protokollbeschreibung

Das Protokoll P9 orientiert sich an den Protokollen P2 bis P4 aus Abschnitt 3.1. Der wesentliche Vorteil gegenüber diesen Protokollen ohne Hardwareunterstützung besteht darin, daß die mit P9 ausgetauschten Objekte weder Generierbarkeit noch Widerrufbarkeit besitzen müssen.

Wie bisher üblich wird zum Aushandeln die Modulimplementierung I1 verwendet. Die Notation $E_V(r)$ steht für die asymmetrische Verschlüsselung von r mit dem öffentlichen

Schlüssel des Vermittlers, während $e_r(O_K)$ die symmetrische Verschlüsselung von O_K mit dem Schlüssel r beschreibt.

Implementierung I2-Hw-term: Zum Vorbereiten des Austausches in I2-Hw-term (siehe Tabelle 5.11) muß der Kunde sein O_K und die Beschreibungen für O_K und O_H an die Chipkarte senden. Diese überprüft O_K und bricht den Austausch ab, wenn sie einen Fehler entdeckt. Ansonsten wählt sie einen zufälligen symmetrischen Schlüssel r, verschlüsselt damit O_K und verwendet den öffentlichen Schlüssel des Vermittlers zum Verschlüsseln von r. Das Ergebnis $E_V(r)$ und $e_r(O_K)$ schickt die Chipkarte dann an den Händler, der diese Werte akzeptiert, wenn sie nachweislich von der Chipkarte berechnet wurden. Falls der Händler keine oder eine fehlerhafte Nachricht von der Chipkarte empfängt, wird er den Austausch einfach beenden, da er noch keine Information über sein Objekt O_H herausgegeben hat. Andernfalls schickt der Händler sein Objekt O_H an die Chipkarte, die dieses Objekt mit der Beschreibung vergleicht. Falls ein Fehler entdeckt wird, muß der Austausch mit I5-Hw-term abgebrochen werden. Ansonsten folgt der Austausch mit I3-Hw-term.

Implementierung I3-Hw-term: Die Chipkarte liefert in I3-Hw-term (siehe Tabelle 5.12) O_H an den Kunden und r an den Händler, womit dieser O_K entschlüsseln kann. Falls dabei ein Fehler auftritt oder der Händler keine Nachricht von der Chipkarte erhält, muß der Händler den Austausch mit I4-Hw-term fortsetzen.

Implementierung I4-Hw-term: Der Händler kann die Konfliktlösung mit I4-Hw-term (siehe Tabelle 5.13) starten, wenn er nach dem Senden von O_H keine oder eine fehlerhafte Antwort von der Chipkarte bekommt. Dazu schickt er dem Vermittler sein Objekt O_H, die Informationen zum Generieren von O_K und die dazugehörigen Beschreibungen. Falls der Austausch bereits mit I5-Hw-term abgebrochen wurde, teilt der Vermittler dies dem Händler mit. Ansonsten prüft er zuerst, ob der Händler ein korrektes O_H geliefert hat. Nur wenn das der Fall ist, entschlüsselt der Vermittler O_K. Falls das Entschlüsseln scheitert, ist dies auf jeden Fall vom Händler verschuldet, da die von der Chipkarte gelieferten Informationen zum Generieren von O_K aufgrund der Vertrauenswürdigkeit der Chipkarte immer korrekt sind. Falls der Vermittler also O_K entschlüsseln kann, speichert er O_H und schickt O_K (oder einfach r) an den Händler.

Implementierung I5-Hw-term: Der Kunde kann mit I5-Hw-term (siehe Tabelle 5.14) den Abbruch des Austausches veranlassen. Falls die Ware bereits ausgeliefert wurde, ist es für ein Zurücksetzen des Austausches zu spät. Ansonsten wendet sich die Chipkarte mit dem Wunsch nach Abbruch des Austausches an den Vermittler. Falls der Händler die Konfliktlösung mit I4-Hw-term bereits erfolgreich durchgeführt hat, liefert der Vermittler das Objekt O_H. Andernfalls wird der Vermittler den Austausch abbrechen und informiert den Kunden darüber.

I2-Hw-term: Implementierung von M2 mit T2-Terminierung für K und H		
K $\rightarrow$ C	:	O_K und Beschreibungen der Objekte
C	:	Prüfe, ob O_K der Beschreibung entspricht (falls nein, beende Austausch)
		Verschlüssele O_K für den Vermittler: $E_V(r)$, $e_r(O_K)$
C $\rightarrow$ H	:	$E_V(r)$, $e_r(O_K)$
H	:	Prüfe diese Werte (falls fehlerhaft, beende Austausch)
H $\rightarrow$ C	:	O_H
C	:	Prüfe, ob O_H der Beschreibung entspricht (falls nein, starte I5-Hw-term)

Tabelle 5.11: Modul M2 für hardwareunterstützten optimistisch fairen Austausch, der beiden Parteien Terminierung garantiert.

I3-Hw-term: Implementierung von M3 mit T2-Terminierung für K und H		
C $\rightarrow$ K	:	O_H
C $\rightarrow$ H	:	r
H	:	Entschlüssele O_K (falls fehlerhaft, starte I4-Hw-term)

Tabelle 5.12: Modul M3 für hardwareunterstützten optimistisch fairen Austausch, der beiden Parteien Terminierung garantiert.

I4-Hw-term: Implementierung von M4 mit T2-Terminierung für K und H		
H $\rightarrow$ V	:	O_H, $E_V(r)$, $e_r(O_K)$, Beschreibungen der Objekte
V	:	Falls K den Austausch bereits abgebrochen hat:
		V $\rightarrow$ H : Austausch abgebrochen
		Sonst:
		Falls H das von K gewünschte O_H geliefert hat:
		Falls der Vermittler ein korrektes O_K entschlüsseln kann:
		V : Speichere O_H
		V $\rightarrow$ H : O_K (oder nur r)
		Falls das Entschlüsseln von O_K fehlschlägt:
		V $\rightarrow$ H : Fehler: Falsches $E_V(r)$, $e_r(O_K)$
		Falls H nicht das von K gewünschte O_H geliefert hat:
		V $\rightarrow$ H : Fehler: Falsches O_H

Tabelle 5.13: Mit dieser Implementierung von Modul M4 kann der Händler immer die Terminierung des Austausches erzwingen.

I5-Hw-term: Implementierung von M5 mit T2-Terminierung für K und H		
K $\rightarrow$ C	:	Austausch abbrechen
C	:	Falls O_H bereits ausgeliefert wurde, beende Konfliktlösung
C $\rightarrow$ V	:	Austausch abbrechen
V	:	Falls I4-Hw-term bereits erfolgreich ausgeführt wurde:
		V $\rightarrow$ K : O_H
		Sonst:
		V $\rightarrow$ K : Austausch abgebrochen

Tabelle 5.14: Diese Implementierung von Modul M5 ermöglicht dem Kunden immer die Terminierung des Austausches.

5.5.2 Nachweis der Protokolleigenschaften

Im folgenden werden die einzelnen Fairneßeigenschaften des Protokolls P9 diskutiert:

Wirksamkeit: Die Wirksamkeit ist erfüllt, da Kunde und Händler im fehlerfreien Fall die gewünschten Objekte in I3-Hw-term erhalten.

Terminierung: Der Kunde terminiert entweder in I3-Hw-term oder im Konfliktlösungsmodul I5-Hw-term. In beiden Fällen ändert sich der Zustand des Kunden nicht mehr, so daß das Protokoll T2-Terminierung für den Kunden sicherstellt.
Der Händler terminiert in I2-Hw-term, I3-Hw-term oder I4-Hw-term. In allen drei Fällen ändert sich der Zustand des Händlers nicht mehr, so daß das Protokoll auch T2-Terminierung für den Händler gewährleistet.

Fairneß: Wenn der Kunde mit O_H terminiert, ändert sich dies nicht mehr, so daß dieser Fall fair ist. Wenn der Kunde ohne O_H terminiert, dann hat er vorher erfolgreich I5-Hw-term ausgeführt. Dann kann der Händler weder in I3-Hw-term noch in I4-Hw-term O_K erhalten, so daß das Protokoll für den Kunden fair ist.
Wenn der Händler mit O_K terminiert, ändert sich dies nicht mehr, so daß dieser Fall fair ist. Wenn der Händler ohne O_K terminiert, dann kann dies in I2-Hw-term oder I4-Hw-term geschehen. Im beiden Fällen kann der Kunde weder im Besitz von O_H sein noch mit I4-Hw-term in dessen Besitz gelangen. Daher ist das Protokoll auch fair für den Händler.
Insgesamt sichert das Protokoll F4-Fairneß für beide Parteien, da die Konfliktlösung mit Hilfe des Vermittlers automatisiert durchgeführt wird.

Nichtabstreitbarkeit: Die Nichtabstreitbarkeit ist in der beschriebenen Version des Austauschprotokolls nicht berücksichtigt. Die Ergänzungen für Nichtabstreitbarkeit der Herkunft und des Empfangs aus Abschnitt 3.2 können dieses Protokoll jedoch problemlos um diese Eigenschaften erweitern.

5.5.3 Diskussion

Die Praktikabilität von Protokoll P9 läßt sich noch steigern, wenn man mit den in Abschnitt 5.3 vorgestellten Ideen die Anforderungen an die Chipkarte reduziert. In I2-Hw-term könnte die Chipkarte das Überprüfen von O_H an den Kunden delegieren, ohne daß sich dadurch Nachteile ergeben. Auch das in Abschnitt 5.3.1 beschriebene externe Speichern von Objekten läßt sich dann auf O_H anwenden. Die einzigen aufwendigeren Aktionen der Chipkarte sind dann das Überprüfen und Verschlüsseln von O_K.

Im Gegensatz zu den bisher in diesem Kapitel vorgestellten Protokollen, die besonders gut für den Austausch von Geld gegen Ware geeignet sind, handelt es sich bei Protokoll P9 um ein sehr generisch einsetzbares Protokoll. Es vereint die Effizienz der optimistischen Protokolle mit der Flexibilität der aktiven Protokolle. Man könnte P9 auch als ein ideales optimistisches Protokoll bezeichnen, weil es F4-Fairneß und T2-Terminierung für beide Parteien gewährleistet und dabei mit beliebigen digitalen Objekten arbeitet.

Eine Chipkarte, die dieses Protokoll implementiert, könnte daher als universelles Hilfsmittel für effizienten fairen Austausch eingesetzt werden und könnte dabei problemlos in die von FlexiFair bereitgestellte Infrastruktur integriert werden. Anstatt die optimistischen Protokolle P2 bis P5 einzusetzen, könnte dann auf Protokoll P9 zurückgegriffen werden, wenn die Eigenschaft der Generierbarkeit oder Widerrufbarkeit nur mit großem Aufwand realisiert werden kann. Auch das aktive Protokoll P1 könnte durch P9 ersetzt werden. Daher handelt es sich bei Protokoll P9 um eine positive Antwort auf die von Asokan formulierte Frage [Aso98, S.137], ob sich beliebige digitale Objekte optimistisch fair austauschen lassen.

Auch aus theoretischer Sicht ist der Vergleich von Protokoll P9 mit den anderen optimistischen Protokollen P2 bis P5 interessant: In [PSW98] wird nachgewiesen, daß beim optimistisch fairen Vertragsaustausch das Protokoll P2 eine minimale Anzahl von Nachrichten (genau 4 Nachrichten zwischen den Parteien in den Modulen M2 und M3) versendet. Die Protokolle P3 bis P5 benötigen zwar weniger Nachrichten, aber sie eignen sich wegen der benötigten Widerrufbarkeit nicht zum Austausch von Verträgen. Das Protokoll P9 dagegen kann dieses bewiesenermaßen optimale Ergebnis aus [PSW98] noch unterbieten, da es in den Modulen M2 und M3 nur 3 Nachrichten zwischen den Parteien verschickt. Der Einsatz von sicherer Hardware ermöglicht also eine deutliche Effizienzsteigerung bei den Austauschprotokollen.

5.6 Zusammenfassung

In diesem Kapitel habe ich neue Fairneßprotokolle entworfen, die fairen Austausch mit sicherer Hardware realisieren. Diese Protokolle besitzen Eigenschaften, die ohne Hardware nicht zu erreichen sind:

- Die Protokolle P7 und P8 können verderbliche Waren fair gegen Geld austauschen. Ohne Hardwareunterstützung könnte ein Wertverlust der verderblichen Ware zu einer Benachteiligung des Empfängers führen.
- Protokoll P9 kann optimistisch fairen Austausch mit beliebigen digitalen Objekten durchführen. Daher werden mit sicherer Hardware grundsätzlich keine aktiven Protokolle benötigt. Dagegen wurde bereits in Kapitel 3 festgestellt, daß alle bekannten optimistischen Protokolle ohne Hardware die Eigenschaft der Generierbarkeit oder Widerrufbarkeit voraussetzen.

Die in Abschnitt 5.3 beschriebenen Optimierungen zeigen, daß die Hardwareunterstützung für fairen Austausch selbst dann praktikabel ist, wenn es sich bei der Hardware um eine Chipkarte mit begrenzter Rechenleistung und Speicher handelt. Durch die externe Speicherung von Daten und das Delegieren der Objektüberprüfungen an Parteien mit mehr Rechenleistung können Protokolle speziell für leistungsschwache Chipkarten optimiert werden. Ein gutes Beispiel für diese Optimierungen stellt das in Abschnitt 5.4.4 beschriebene Protokoll P8 dar.

Die in diesem Kapitel vorgeschlagene Idee der Hardwareunterstützung für fairen Austausch läßt sich in verschiedenen Szenarien besonders gut praktisch einsetzen: Bei einem

elektronischen Einkauf wird oft eine Chipkarte zur Durchführung der Bezahlung verwendet, so daß die Hardwareanforderungen bereits erfüllt sind. Außerdem stellt der Einsatz von sicherer Hardware die einzige bisher bekannte Möglichkeit dar, wie sich der Einkauf von verderblichen Waren fair abwickeln läßt. Ein anderes Szenario, in dem Chipkarten bereits heute eingesetzt werden, ist die elektronische Vertragsunterzeichnung. Auch hier könnte zusätzliche Funktionalität für fairen Austausch die Einsatzmöglichkeiten dieser Karte erweitern.

In beiden Fällen ist die Austauschfunktionalität nicht auf die in diesen Szenarien ausgetauschten Objekte begrenzt. Eine Karte mit Bezahlfunktion könnte z.B. auch für fairen Austausch einer E-Mail gegen eine Empfangsbestätigung verwendet werden, wenn die Chipkarte ein generisches Protokoll wie P9 unterstützt. Mit diesem effizienten und vielseitigen Protokoll kann eine Chipkarte als eine Art „FlexiFair-Karte“ eingesetzt werden, da sie viele unterschiedliche Anwendungen beim fairen Austausch beliebiger digitaler Objekte unterstützen kann.

6 Zusammenfassung

In dieser Arbeit habe ich mich mit den theoretischen und praktischen Problemen beschäftigt, die bei der Realisierung einer Infrastruktur für fairen Austausch gelöst werden müssen. Das Ziel einer solchen Infrastruktur besteht darin, daß nicht mehr jede Anwendung eine eigene Lösung für fairen Austausch entwickeln muß, sondern auf generische Lösungen zurückgreifen kann, die von einem Fairneßdienst bereitgestellt werden. Eine Anwendung profitiert dann von einem solchen Fairneßdienst dadurch, daß die Bündelung von Austauschfunktionalität die Kosten senkt und die Sicherheit erhöht. Daher habe ich hier gezeigt, wie ein solcher Fairneßdienst entworfen und implementiert werden kann.

Damit eine Anwendung die Eigenschaften eines vom Fairneßdienst FlexiFair unterstützten Austauschprotokolls einschätzen kann, müssen dessen Eigenschaften genau spezifiziert sein. Daher habe ich eine Fairneßhierarchie vorgeschlagen, die die Fairneßeigenschaften von Protokollen präziser beschreibt, als das andere bisher bekannte Fairneßdefinitionen erlauben. Insbesondere die fein abgestuften Grade der Fairneß und Terminierung ermöglichen die bessere Differenzierung zwischen verschiedenen Austauschprotokollen und erleichtern so die Auswahl eines vom Fairneßdienst FlexiFair bereitgestellten Protokolls.

Da es eine Vielzahl verschiedener Fairneßprotokolle gibt, die ein Fairneßdienst bereitstellen kann, habe ich eine generische Modellierung aller aktiven und optimistischen Protokolle entwickelt, die F4-Fairneß garantieren. Durch diese modulare Modellierung kann auf alle Protokolle über eine einheitliche Schnittstelle zugegriffen werden. Andere bisher veröffentlichte Ansätze liefern höchstens eine generische Beschreibung für optimistische Protokolle, die Generierbarkeit zur Konfliktlösung verwenden. Ein generischer Fairneßdienst muß jedoch unbedingt auch aktive Protokolle unterstützen, da diese die geringsten Anforderungen an die ausgetauschten Objekte stellen.

Die Leistungsfähigkeit dieser Modellierung von Austauschprotokollen haben ich in Kapitel 3 unter Beweis gestellt, indem ich diese Modellierung systematisch auf verschiedene Fairneßprotokolle angewendet habe. Neben den aktiven Protokollen und den optimistischen Protokollen, die Generierbarkeit zur Konfliktlösung verwenden, habe ich mein Hauptaugenmerk dabei speziell auf die Klasse der optimistischen Protokolle gerichtet, die Widerrufbarkeit zur Konfliktlösung einsetzen. Mit den Protokollen P3 und P4 ist es mir gelungen, neue Konstruktionsprinzipien für optimistisch fairen Austausch zu entwickeln. Dagegen war bisher nur das auf Generierbarkeit basierende Konstruktionsprinzip für optimistisch fairen Austausch bekannt.

Durch die systematische Untersuchung verschiedener Austauschprotokolle und die Diskussion von Erweiterungen für Nichtabstreitbarkeit der Herkunft und des Empfangs gebe ich eine umfassende Übersicht über alle wesentlichen Austauschprotokolle, die F4-Fairneß

sicherstellen. Anhand der Vorbedingungen und der von mir nachgewiesenen Fairneßeigenschaften kann jetzt eine Anwendung leicht aus diesen verschiedenen Protokollen ein besonders gut geeignetes heraussuchen.

Bevor ich allerdings einen Prototyp für meinen Fairneßdienst entwickeln konnte, mußte ich auch auf der Implementierungsebene verschiedene Probleme lösen. Da ein Vermittler besonders vertrauenswürdig sein muß, sollte dessen Implementierung sehr sorgfältig durchgeführt werden, weshalb im laufenden Betrieb möglichst wenige Änderungen an dessen Software vorgenommen werden sollten. Allerdings hat FlexiFair auch das Ziel, viele verschiedene Anwendungen zu unterstützen. Sobald eine Anwendung die bereits vorinstallierten Klassen für die auszutauschenden Objekte und ihre Beschreibungen nicht verwenden kann, muß allerdings die Funktionalität des Vermittlers erweitert werden. Da dies einen großen Aufwand bedeuten kann, habe ich einen dynamischen Ansatz zur Bereitstellung der benötigten Vermittlerfunktionalität entwickelt. Bei aktiven Protokollen erlaube ich den Anwendungen, eigene Objektbeschreibungen zu liefern und so die gewünschte Funktionalität selbst mitzubringen. Die Sicherheit wird bei diesem Ansatz durch die Begrenzung der Ausführungsrechte der fremden Programmteile garantiert.

Weitere Sicherheitsfragen müssen bei der Implementierung gelöst werden, damit ein Angreifer die Protokollausführung nicht manipulieren kann. Durch eine aus verschiedenen Protokollparametern berechnete Transaktionsnummer stelle ich eine Verknüpfung zwischen den einzelnen Nachrichten eines Protokolls her. Daher kann jede Art der Veränderung und auch das Wiedereinspielen von alten Nachrichten entdeckt und somit verhindert werden.

Schließlich zeige ich anhand von verschiedenen Beispielen, daß Anwendungen in sehr unterschiedlichen Szenarien die von FlexiFair bereitgestellte Austauschfunktionalität nutzen können. Die besondere Flexibilität von FlexiFair zeigt sich dadurch, daß auch ein von mir neu vorgeschlagenes Austauschszenario problemlos durch die von FlexiFair angebotenen Dienste unterstützt wird.

Im Gegensatz zu den rein in Software implementierten Fairneßlösungen ist in Zukunft auch eine teilweise Implementierung von Austauschprotokollen in Hardware denkbar. Die immer schneller und trotzdem preiswerter werdenden Chipkarten bieten dabei ein gutes Mittel zur Unterstützung von fairem Austausch. Daher habe ich in dieser Arbeit erstmals die Möglichkeiten von Hardwareunterstützung für fairen Austausch untersucht. Ein Ergebnis besteht darin, daß auf diese Weise auch bisher nicht fair austauschbare verderbliche Waren wie Echtzeitbörseninformationen fair ausgetauscht werden können. Als ein weiteres Ergebnis zeigt das Protokoll P9 das Potential von Hardwareunterstützung. Dieses Protokoll vereint die Vorteile von aktiven und optimistischen Protokollen. Es kann beliebige digitale Objekte austauschen und trotzdem handelt es sich um ein besonders effizientes optimistisches Protokoll. Aus diesen Gründen stellt eine Hardwareunterstützung eine ideale Ergänzung eines Fairneßdienstes wie FlexiFair dar.

Mit dieser Arbeit ermögliche ich durch die Erweiterung der theoretischen Grundlagen ein besseres Verständnis für fairen Austausch. Durch die Implementierung habe ich gezeigt, wie sich fairer Austausch flexibel, effizient und kostengünstig realisieren läßt. Schließlich habe ich mit der Hardwareunterstützung einen Ausblick auf mögliche zukünf-

tige Entwicklungen gegeben. Diese Arbeit kann darum als Beitrag zur Verbesserung der Praxistauglichkeit von fairem Austausch angesehen werden.

A Die Java-Klassen von FlexiFair

Dieser Anhang gibt einen Überblick über die wesentlichen Java-Klassen von FlexiFair. Von jeder Klasse sind nur die öffentlichen Methoden und Attribute angegeben.

A.1 Die ausgetauschten Objekte

A.1.1 Basisklassen

Alle ausgetauschten Objekte sind von der Klasse `ExchangeableItem` abgeleitet. Zusätzlich können Objekte noch die vier Eigenschaften `Generatable`, `Revocable`, `NRO` oder `NRR` besitzen, die in der Form von Interfaces definiert sind.

```
package flexifair.item;
public class ExchangeableItem implements Serializable {
   public boolean isGeneratable = (this instanceof Generatable);
   public boolean isRevocable = (this instanceof Revocable);
   public boolean NRO = (this instanceof NRO);
   public boolean NRR = (this instanceof NRR);
   public ExchangeableItem();
   public ExchangeableItem(Serializable value);
   public void setValue(Serializable value);
   public Serializable getValue();
}

package flexifair.item.properties;
public interface Generatable extends Exchangeable {
   public void setGenerateInfo(Serializable geninfo);
   public Serializable getGenerateInfo();
   public boolean generateItemByTTP(GeneratableItemDescription desc,
          Serializable geninfo, GenerateSecret gensecret, TransactionID tid);
}

package flexifair.item.properties;
public interface Revocable extends Exchangeable {
   public boolean revokeItemByTTP(Object secretTTPinfo);
}
```

```
package flexifair.item.properties;
public interface NRO {
   public boolean NROavailable();
   public byte[] getNRO();
   public void setNRO(byte[] nro_signature);
}

package flexifair.item.properties;
public interface NRR {
   public boolean NRRavailable();
   public NRRToken getNRR();
   public void setNRR(NRRToken nrr_token);
}
```

A.1.2 Generierbare Objekte

Generierbare Objekte implementieren das Interface `Generatable`. Der Vermittler generiert ein solches Objekt mit der Generierungsinformation und einer geheimen Zusatzinformation vom Typ `GenerateSecret` . Ein Beispiel für ein generierbares Objekt ist die Klasse `GenContract`. Der Vermittler kann mit der aus einer Signatur bestehenden Generierungsinformation und der geheimen Zusatzinformation `ContractGenerateSecret` einen Vertrag vom Typ `ReplacementContractValue` erzeugen.

```
package flexifair.item;
public abstract class GeneratableItem extends ExchangeableItem
                                       implements Generatable {
   public GeneratableItem();
   public GeneratableItem(Serializable value);
   public void setGenerateInfo(Serializable geninfo);
   public Serializable getGenerateInfo();
}

package flexifair.item;
public class GenContract extends GeneratableItem {
   public GenContract();
   public GenContract(Serializable value);
   public GenContract(PrivateKey priv, String algo_name, byte[] contract);
   public boolean setGenerateInfo(TransactionID tid, PrivateKey priv,
          String algo_name, byte[] intent, byte[] contract);
   public boolean generateItemByTTP(GeneratableItemDescription desc,
          Serializable geninfo, GenerateSecret gensecret,
          TransactionID tid);
}

package flexifair.item.ttp;
```

```
public class ContractGenerateSecret implements GenerateSecret {
   public ContractGenerateSecret(PrivateKey ttp_private_key,
          String ttp_sig_algo);
   public ReplacementContractValue issueReplacementContract(byte[] geninfo);
}

package flexifair.item.ttp;
public interface GenerateSecret extends Secret {
}

package flexifair.item.ttp;
public interface Secret {
}

package flexifair.item;
public class ReplacementContractValue implements Serializable {
   public ReplacementContractValue(byte[] intent_sig, byte[] replacement_sig);
   public byte[] getIntentSig();
   public byte[] getReplacementSig();
}
```

A.1.3 Widerrufbare Objekte

Widerrufbare Objekte implementieren das Interface `Revocable`. Der Vermittler kann mit geheimen Zusatzinformationen vom Typ `RevokeSecret` solche Objekte widerrufen.

```
package flexifair.item;
public abstract class RevocableItem extends ExchangeableItem
                                     implements Revocable {
   public RevocableItem();
   public RevocableItem(Serializable item);
}

package flexifair.item.ttp;
public interface RevokeSecret extends Secret {
}
```

A.1.4 Nichtabstreitbarkeit

Wenn mit einem Objekt auch Nachweise des Empfangs oder der Herkunft ausgetauscht werden sollen, muß dieses Objekt das Interface `NRO` bzw. `NRO` implementieren. Beispiele dafür sind die Klassen `NROItem`, `NRRItem` und `NRORItem`. Ein Nachweis der Herkunft besteht einfach aus einer Signatur, was in Java ein Byte-Array ist. Ein Nachweis des

Empfangs wird durch die Klasse `NRRToken` repräsentiert und enthält entweder die Signatur des Empfängers oder die Ersatzsignatur des Vermittlers.

```
package flexifair.item;
public class NROItem extends ExchangeableItem implements NRO {
   public NROItem();
   public NROItem(Serializable value);
   public NROItem(Serializable value, byte[] nro_sig);
   public boolean NROavailable();
   public void setNRO(byte[] nro_signature);
   public byte[] getNRO();
}

package flexifair.item;
public class NRRItem extends ExchangeableItem implements NRR {
   public NRRItem();
   public NRRItem(Serializable value);
   public boolean NRRavailable();
   public void setNRR(NRRToken nrr_token);
   public NRRToken getNRR();
}

package flexifair.item;
public class NRORItem extends NROItem implements NRR {
   public NRORItem();
   public NRORItem(Serializable value);
   public NRORItem(Serializable value, byte[] nro_sig);
   public boolean NRRavailable();
   public void setNRR(NRRToken nrr_token);
   public NRRToken getNRR();
}

package flexifair.item;
public class NRRToken implements Serializable {
   public NRRToken(byte[] nrr_sig, boolean fromParticipant);
   public void setToken(byte[] nrr_sig, boolean fromParticipant);
   public byte[] getToken();
   public boolean getFromParticipant();
}
```

A.2 Die Beschreibungen der Objekte

Alle Beschreibungen von Objekten müssen das Interface `Description` implementieren. Bei generierbaren Objekten müssen zusätzlich die in `GeneratableItemDescription`

definierten Methoden bereitgestellt werden. Ein Beispiel für eine solche Beschreibung eines generierbaren Objekts ist die Klasse `GenContractDescription`. Andere Beispiel sind die Klassen `NRODescription` und `NRRDescription`.

```
package flexifair.item.description;
public interface Description extends Serializable {
   public String toString();
   public boolean verifyItem(ExchangeableItem item, TransactionID tid);
   public boolean isGeneratableItem();
   public boolean isRevocableItem();
   public String getDefaultItemName();
}

package flexifair.item.description;
public interface GeneratableItemDescription extends Description {
   public boolean verifyGenInfo(Serializable geninfo, TransactionID tid);
   public String getGenSecretName();
   public boolean generatedByTTP();
}

package flexifair.item.description;
public class GenContractDescription implements GeneratableItemDescription {
   public GenContractDescription(PublicKey pub_key_client,
          String algorithm_client, PublicKey pub_key_ttp, String algorithm_ttp,
          byte[] plaintext, byte[] intent_text);
   public boolean isGeneratableItem();
   public boolean isRevocableItem();
   public String toString();
   public boolean verifyItem(ExchangeableItem item, TransactionID tid);
   public boolean generatedByTTP();
   public boolean verifyGenInfo(Serializable obj, TransactionID tid);
   public String getGenSecretName();
   public String getTTPSigAlgo();
   public String getDefaultItemName();
}

package flexifair.item.description;
public class NRODescription implements Description {
   public NRODescription(PublicKey pub_key, String sig_algo);
   public String toString();
   public boolean verifyItem(ExchangeableItem item, TransactionID tid);
   public boolean isGeneratableItem();
   public boolean isRevocableItem();
   public String getDefaultItemName();
}
```

```
package flexifair.item.description;
public class NRRDescription implements Description {
   public NRRDescription(PublicKey pub_key_participant,
          String sig_algo_participant, PublicKey pub_key_TTP,
          String sig_algo_TTP);
   public String toString();
   public boolean verifyItem(ExchangeableItem item, TransactionID tid);
   public boolean isGeneratableItem();
   public boolean isRevocableItem();
   public String getDefaultItemName();
}
```

Da FlexiFair auch durch selbstdefinierte Beschreibungen erweitert werden kann, werden Beschreibungen nicht direkt an andere Parteien geschickt. Stattdessen werden sie in der Klasse `DescriptionContainer` mit einer eventuell vorhandenen Jar-Datei zusammengefaßt, die den ausführbaren Java-Code der neugeschriebenen Klassen enthält. Beim Versenden eines Objekts vom Typ `DescriptionContainer` werden dann alle nicht beim Vermittler installierte Klassen aus der Jar-Bibliothek geladen. Damit diese Klassen mit reduzierten Zugriffsrechten ausgeführt werden, muß das Nachladen mit der Klasse `JarClassLoader` erfolgen, die von `SecureClassLoader` erbt.

```
package flexifair.io;
public class DescriptionContainer implements Serializable {
   public DescriptionContainer(Description desc, byte[] jarbytes);
   public void setDescription(Description desc);
   public Description getDescription();
   public void setJarBytes(byte[] bytes);
   public byte[] getJarBytes();
   private void writeObject(ObjectOutputStream s) throws IOException;
   private void readObject(ObjectInputStream s) throws IOException,
                                                ClassNotFoundException;
}

package flexifair.util;
public class JarClassLoader extends SecureClassLoader {
   public JarClassLoader();
   public JarClassLoader(ClassLoader parent);
   public JarClassLoader(byte[] bytes);
   public JarClassLoader(byte[] bytes ,ClassLoader parent);
   public void init(byte[] bytes);
   public void init(String filename) throws IOException;
   public Class loadClass(String name) throws ClassNotFoundException;
   public Class findClass(String name) throws ClassNotFoundException;
}
```

A.3 Die Austauschprotokolle

Alle Austauschprotokolle sind von der Klasse `ExchangeProtocol` abgeleitet. Diese definiert für die am Austausch beteiligten Parteien und den Vermitter die Methoden, mit denen ein fairer Austausch durchgeführt werden kann. Die als abstrakt definierten Methoden werden zum Beispiel von den Klassen `ActiveProtocol` , `SynchronousOptimisticProtocol` oder `ASW98Protocol` implementiert.

```
package flexifair.protocol;
public abstract class ExchangeProtocol implements Serializable {
   public void initClient(TransactionID transaction_id, PrivateKey priv_key,
          boolean isOriginator, Storage storage) throws ExchangeException;
   public void initTTP(Storage storage, TransactionID transaction_id)
          throws ExchangeException;
   public void setTTPSecrets(TTPSecrets secrets);
   public static ExchangeProtocol getInstance(String algorithm);
   public abstract boolean needsGeneratablity(boolean originator);
   public abstract boolean needsRevocability(boolean originator);
   public abstract boolean canResolve(boolean originator);
   public abstract boolean canAbort(boolean originator);
   public abstract boolean prepareExchangeClient(ExchangeableItem own_item,
          Connection client_con, Connection ttp_con);
   public abstract ExchangeableItem doExchangeClient(Connection client_con,
          Connection ttp_con);
   public abstract ExchangeableItem resolveExchangeClient(Connection client_con,
          Connection ttp_con);
   public abstract boolean abortExchangeClient(Connection client_con,
          Connection ttp_con);
   public abstract void prepareExchangeTTP(Connection client_con);
   public abstract void doExchangeTTP(Connection client_con);
   public abstract void resolveExchangeTTP(Connection client_con);
   public abstract void abortExchangeTTP(Connection client_con);
}
```

Die Kommunikation aller Austauschprotokolle erfolgt immer über eine Implementierung von `Connection`. Ein Beispiel ist die Klasse `StreamConnection`, die die Kommunikation über bestehende TCP/IP-Verbindung abwickelt.

```
package flexifair.io;
public interface Connection {
   public boolean isAvailable() throws IOException;
   public void writeObject(Object obj) throws IOException;
   public void writeInt(int i) throws IOException;
   public void writeBoolean(boolean b) throws IOException;
   public Object readObject() throws IOException, ClassNotFoundException;
   public int readInt() throws IOException;
```

```
    public boolean readBoolean() throws IOException;
    public void close() throws IOException;
}

package flexifair.io;
public class StreamConnection implements Connection {
    public StreamConnection(InputStream in, OutputStream out) throws IOException;
    public boolean isAvailable() throws IOException;
    public void writeObject(Object obj) throws IOException;
    public void writeInt(int i) throws IOException;
    public void writeBoolean(boolean b) throws IOException;
    public Object readObject() throws IOException, ClassNotFoundException;
    public int readInt() throws IOException;
    public boolean readBoolean() throws IOException;
    public void close() throws IOException;
}
```

Eine Austauschtransaktion wird immer unter einer bestimmten Transaktionsnummer durchgeführt, die von der Klasse `TransactionID` berechnet wird. Dazu müssen alle die Transaktion identifizierenden Daten übergeben werden.

```
package flexifair.protocol;
public class TransactionID implements Serializable {
    public TransactionID(String fex_algo, String ttp_url, String sig_algo_ttp,
            PublicKey pk_ttp, PublicKey pk_originator, PublicKey pk_recipient,
            String sig_algo_originator, String sig_algo_recipient,
            byte[] nonce_originator, byte[] nonce_recipient,
            byte[] jar_desc_originator, byte[] jar_desc_recipient,
            Description desc_originator_item, Description desc_recipient_item);
    public PublicKey getTTPPublicKey();
    public String getTTPSignatureAlgorithm();
    public PublicKey getPublicKey(boolean originator);
    public String getSignatureAlgorithm(boolean originator);
    public Description getDescription(boolean originator);
    public byte[] getJarBytes(boolean originator);
    public byte[] getNonce(boolean originator);
    public String getExchangeAlgorithm();
    public String getTTPAdress();
    public byte[] getHash();
    public byte[] getHash(MessageDigest md);
    public int hashCode();
    public boolean equals(Object obj);
}
```

Der Zustand von Austauschprotokollen läßt sich mit einer Implementierung der Klasse `Storage` abspeichern und so vor einem Programmabsturz sichern. Ein Beispiel für eine

solche Implementierung ist die Klasse `FileStorage`, bei der die Daten in einer Datei gesichert werden.

```
package flexifair.protocol.persistence;
public abstract class Storage {
   public abstract boolean containsKey(TransactionID key);
   public abstract Serializable get(TransactionID key);
   public abstract boolean put(TransactionID key, Serializable value);
}

package flexifair.protocol.persistence;
public class FileStorage extends Storage {
   public FileStorage();
   public FileStorage(String filename);
   public boolean put(TransactionID key, Serializable value);
   public Serializable get(TransactionID key);
   public boolean containsKey(TransactionID key);
}
```

A.4 Der Vermittler

Die Klasse `TTPServer` stellt die Dienste eines Vermittlers bereit. Nach dem Start wartet dieser Dienst auf Anfragen, für die dann jeweils ein neuer Thread erzeugt wird. Zur dauerhaften Speicherung von Daten greift der Vermittler auf eine Datenbank wie zum Beispiel MySQL zurück.

```
package flexifair.ttp;
public class TTPServer extends Thread implements ProtocolDefinition {
   public static final int TTPPORT = 2000;
   public TTPServer(Socket s);
   public TTPServer(Socket s, String address, int portnbr,
           String jdbc_driver, String db_protocol);
   public void run();
   public static void main (String args[]);
}
```

A.5 Die Beispielapplikation zum Austausch von Verträgen

Mit der `ContractSigning` Applikation können zwei Parteien einen fairen Austausch von digital signierten Verträgen durchführen. Beim Start der Applikation sollte der Name einer Konfigurationsdatei übergeben werden, die die Preferenzen dieser Partei enthält. Die grafische Oberfläche dieser Anwendung ist in Abbildung A.1 dargestellt. Der Nutzer kann verschiedene Voreinstellungen anpassen und insbesondere ein geeignetes Austauschprotokoll auswählen. Bei der Durchführung von fairem Austausch kann der Nutzer die

Bearbeitung der verschiedenen Module manuell anstoßen und so auch zwischen den verschiedenen Arten der Konfliktlösung wählen.

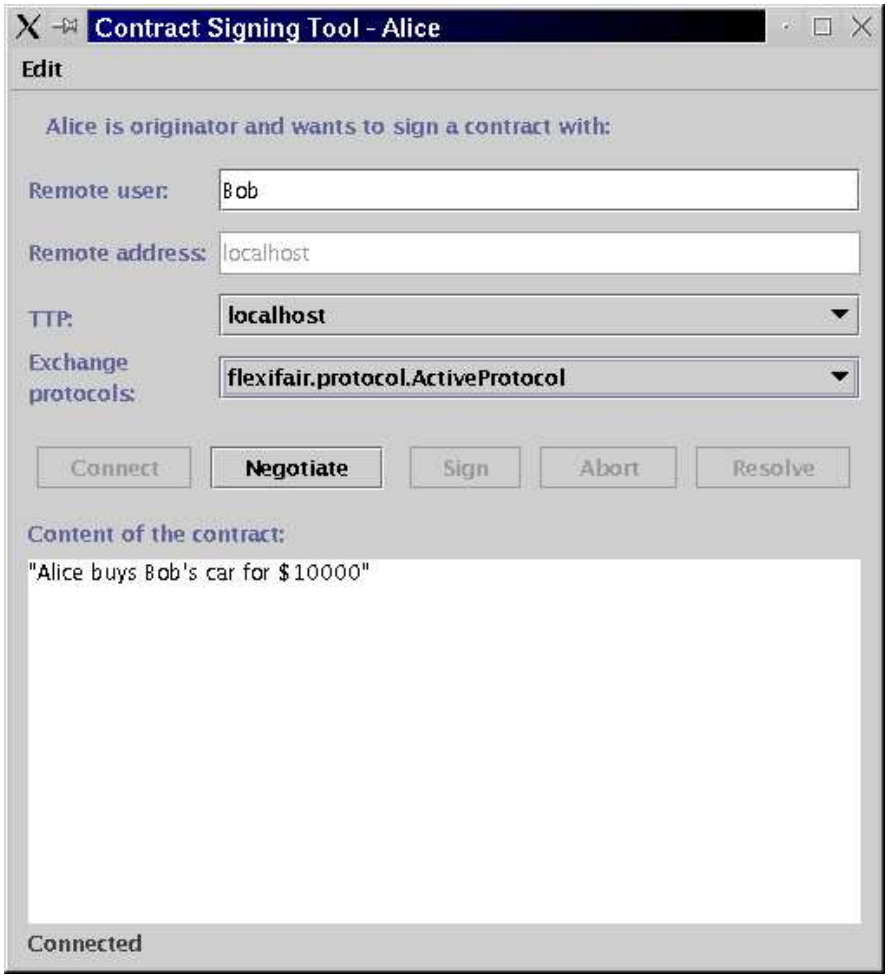

Abbildung A.1: Die Anwendung zur Vertragsunterzeichnung.

```
package flexifair.client;
public class ContractSigning extends JFrame {
   public ContractSigning(String filename);
   public static void main(String[] args);
}
```

Literaturverzeichnis

[Aso98] N. Asokan. *Fairness in electronic commerce*. PhD thesis, University of Waterloo, Canada, May 1998.

[ASW97a] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In Tsutomu Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 6–17, Zürich, Switzerland, April 1997. ACM Press.

[ASW97b] N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. Technical Report RZ 2976 (#93022), IBM Research, November 1997. Full version of [ASW98a].

[ASW97c] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. Technical Report RZ 2973 (#93019), IBM Research, November 1997. Full version of [ASW98b].

[ASW98a] N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, Oakland, CA, May 1998. IEEE Computer Society Press.

[ASW98b] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606, Espoo, Finland, June 1998. Springer-Verlag.

[ASW00] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, April 2000.

[Ate99] Giuseppe Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *Proceedings of 6th ACM Conference on Computer and Communications Security (CCS '99)*, pages 138–146, Singapore, November 1999. ACM Press.

[BCDP90] J. Boyar, D. Chaum, I. B. Damgård, and T. P. Pedersen. Convertible undeniable signatures. In *Advances in Cryptology – CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 189–205. Springer-Verlag, 1990.

[BCDvdG87] Ernest F. Brickell, David Chaum, Ivan B. Damgård, and Jeroen van de Graaf. Gradual and verifiable release of a secret. In *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 156–166, Santa Barbara, CA, August 1987. Springer-Verlag.

[BDM98] Feng Bao, R. H. Deng, and Wenbo Mao. Efficient and practical fair exchange protocols with off-line TTP. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 77–85, Oakland, CA, May 1998. IEEE Computer Society Press.

[BF98] Colin Boyd and Ernest Foo. Off-line fair payment protocol using convertible signatures. In *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 271–285, Beijing, China, October 1998. Springer-Verlag.

[BH01] Levente Buttyán and Jean-Pierre Hubaux. Rational exchange – A formal model based on game theory. In *Electronic Commerce – WELCOM 2001*, volume 2232 of *Lecture Notes in Computer Science*, pages 114–126, Heidelberg, Germany, November 2001. Springer-Verlag.

[BK00] Colin Boyd and Peter Kearney. Exploring fair exchange protocols using specification animation. In *Information Security – ISW 2000*, volume 1975 of *Lecture Notes in Computer Science*, pages 209–223, Wollongong, Australia, December 2000. Springer-Verlag.

[Blu83] Manuel Blum. How to exchange (secret) keys. *ACM Transactions on Computer Systems*, 1(2):175–193, May 1983.

[BN00] Dan Boneh and Moni Naor. Timed commitments. In *Advances in Cryptology – CRYPTO '2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254, Santa Barbara, CA, 2000. Springer-Verlag.

[BNS01] Matthias Baumgart, Heike Neumann, and Niko Schweitzer. Optimistisch faire transaktionen mittels zeitlich beschränkter münzen. In *Verlässliche IT-Systeme – VIS 2001*, pages 125–133, Kiel, Germany, September 2001.

[BOGMR90] Michael Ben-Or, Oded Goldreich, Silvio Micali, and Ronald L. Rivest. A fair protocol for signing contracts. *ACM Transactions on Information Theory*, 36(1):40–46, January 1990.

[BP90] Holger Bürk and Andreas Pfitzmann. Value exchange systems enabling security and unobservability. *Computers & Security*, 9(8):715–721, 1990.

[Bra93] Stefan Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318, Santa Barbara, CA, August 1993. Springer-Verlag.

[BT94] Alireza Bahreman and J.D. Tygar. Certified electronic mail. In *Proceedings of the ISOC Symposium on Network and Distributed Systems Security*, pages 3–19, San Diego, CA, February 1994. IEEE Computer Society Press.

[But00] Levente Buttyán. Removing the financial incentive to cheat in micropayment schemes. *IEE Electronics Letters*, 36(2):132–133, January 2000.

[CD98] Jan Camenisch and Ivan Damgård. Verifiable encryption and applications to group signatures and signature sharing. Technical Report RS-98-32, BRICS, Department of Computer Science, Aarhus University, Denmark, December 1998.

[Cer02] Certifiedmail.com. http://www.certifiedmail.com, 2002.

[Cha83] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology – CRYPTO '82*, pages 199–203. Plenum, 1983.

[Cha90] David Chaum. Zero-knowledge undeniable signatures. In *Advances in Cryptology – EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464. Springer-Verlag, 1990.

[Che98] Liqun Chen. Efficient fair exchange with verifiable confirmation of signatures. In K. Ohta and D. Pei, editors, *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 286–299, Beijing, China, 18–22 October 1998. Springer-Verlag.

[CHTY96] Jean Camp, Michael Harkavy, J. D. Tygar, and Bennet Yee. Anonymous atomic transactions. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pages 123–133, Oakland, CA, November 1996.

[CMS96] Jan Camenisch, Ueli Maurer, and Markus Stadler. Digital payment systems with passive anonymity-revoking trustees. In *Computer Security – ESORICS '96*, volume 1146 of *Lecture Notes in Computer Science*, pages 31–43, Rome, Italy, September 1996. Springer-Verlag.

[CTS95] Benjamin Cox, J.D. Tygar, and Marvin Sirbu. Netbill security and transaction protocol. In *First USENIX Workshop of Electronic Commerce Proceedings*, pages 77–88, New York, July 1995.

[CvA90] David Chaum and Hans van Antwerpen. Undeniable signaures. In *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer-Verlag, 1990.

[Dam93] I. B. Damgård. Practical and provably secure release of a secret and exchange of signatures. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 200–217, Lofthus, Norway, May 1993. Springer-Verlag.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In *Advances in Cryptology – CRYPTO '82*, pages 205–210, New York, USA, 1982. Plenum Publishing.

[Eve81] Shimon Even. A protocol for signing contracts. In Allen Gersho, editor, *Advances in Cryptology: A Report on CRYPTO 81*, pages 148–153, Santa Barbara, USA, August 1981. ECE Report No 82-04, U.C. Santa Barbara, Department of Elec. and Computer Eng.

[EY80] Shimon Even and Yacov Yacobi. Relations amoung public key signature systems. Technical Report 175, Computer Science Department, Technicon, Haifa, Israel, 1980.

[FPH00] Josep L. Ferrer-Gomila, Magdalena Payeras-Capellà, and Llorenç Huguet i Rotger. An efficient protocol for certified mail. In *Information Security – ISW 2000*, volume 1975 of *Lecture Notes in Computer Science*, pages 237–248, Wollongong, Australia, December 2000. Springer-Verlag.

[FPH01] Josep L. Ferrer-Gomila, Magdalena Payeras-Capellà, and Llorenç Huguet i Rotger. Efficient optimistic n-party contract signing protocol. In *Information Security – ISC 2001*, volume 2200 of *Lecture Notes in Computer Science*, pages 394–407, Malaga, Spain, October 2001. Springer-Verlag.

[FR97] Matthew K. Franklin and Michael K. Reiter. Fair exchange with a semi-trusted third party. In Tsutomu Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 1–5, Zürich, Switzerland, April 1997. ACM Press.

[GJM99] Juan A. Garay, Markus Jakobsson, and Philip MacKenzie. Abuse-free optimistic contract signing. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466, Santa Barbara, CA, 15–19 August 1999. Springer-Verlag.

[GPV99] Felix C. Gärtner, Henning Pagnia, and Holger Vogt. Approaching a formal definition of fairness in electronic commerce. In *Proceedings of the International Workshop on Electronic Commerce (WELCOM '99)*, pages 354–359, Lausanne, Switzerland, October 1999. IEEE Computer Society Press.

[GR93] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann, 1993.

[GRV03] Sigrid Gürgens, Carsten Rudolph, and Holger Vogt. On the security of fair non-repudiation protocols. In *Information Security – ISC 2003*, volume 2851 of *Lecture Notes in Computer Science*, pages 193–207, Bristol, UK, October 2003. Springer-Verlag.

[ISO97] ISO/IEC 13888-1, 13888-2, and 13888-3. Non-repudiation. ISO/IEC JTC 1/SC 27, 1997.

[Jak95] Markus Jakobsson. Ripping coins for fair exchange. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology – EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 220–230, St. Malo, France, 21–25 May 1995. Springer-Verlag.

[KM00] Steve Kremer and Olivier Markowitch. Optimistic non-repudiable information exchange. In J. Biemond, editor, *21st Symposium on Information Theory in the Benelux*, pages 139–146, Wassenaar, The Netherlands, May 2000. Werkgemeenschap Informatieen Communicatietheorie, Enschede.

[KM01] Steve Kremer and Olivier Markowitch. Selective receipt in certified e-mail. In *Progress in Cryptology – INDOCRYPT 2001*, volume 2247 of *Lecture Notes in Computer Science*, pages 136–148, Chennai, India, 16–20 December 2001. Springer-Verlag.

[KMZ02] Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, November 2002.

[KR01] Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. In *CONCUR 2001 – Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, pages 551–565, Aalborg, Denmark, August 2001. Springer-Verlag.

[Küg02] Dennis Kügler. *Ein mißbrauchfreies anonymes elektronisches Zahlungssystem.* PhD thesis, Darmstadt University of Technology, August 2002.

[KV01a] Dennis Kügler and Holger Vogt. Marking: A privacy protecting approach against blackmailing. In *Public Key Cryptography – PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 137–152, Cheju Island, Korea, February 2001. Springer-Verlag.

[KV01b] Dennis Kügler and Holger Vogt. Fair tracing without trustees. In *Financial Cryptography – FC 2001*, volume 2339 of *Lecture Notes in Computer Science*, pages 136–148, Grand Cayman, BWI, February 2001. Springer-Verlag.

[KV01c] Dennis Kügler and Holger Vogt. Unsichtbare Markierungen in elektronischem Geld. In *Kommunikationssicherheit – Schwerpunkt Internet*, pages 262–271. Vieweg Verlag, March 2001.

[KV01d] Dennis Kügler and Holger Vogt. Auditable tracing with unconditional anonymity. In *Proceedings of the 2nd International Workshop on Information Security Applications (WISA 2001)*, pages 151–163, Seoul, Korea, September 2001.

[KV02] Dennis Kügler and Holger Vogt. Off-line payments with auditable tracing. In *Financial Cryptography – FC 2002*, volume 2357 of *Lecture Notes in Computer Science*, pages 269–281, Southampton, Bermuda, March 2002. Springer-Verlag.

[LNJ00] Peng Liu, Peng Ning, and Sushil Jajodia. Avoiding loss of fairness owing to process crashes in fair data exchange protocols. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks, Workshop on Dependability despite Malicious Faults*, pages 631–640, New York, June 2000. IEEE Computer Society Press.

[Lou00] Panagiotis Louridas. Some guidelines for non-repudiation protocols. *Computer Communication Review*, 30(5):29–38, October 2000.

[LPSW00] Gérard Lacoste, Birgit Pfitzmann, Michael Steiner, and Michael Waidner, editors. *SEMPER – Secure Electronic Marketplace for Europe*, volume 1854 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[Lyn96] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[Mao97] Wenbo Mao. Verifiable escrowed signature. In *Information Security and Privacy – ACISP '97*, volume 1270 of *Lecture Notes in Computer Science*, pages 240–248, Sydney, Australia, July 1997. Springer-Verlag.

[MGK02] Olivier Markowitch, Dieter Gollmann, and Steve Kremer. On fairness in exchange protocols. In *Information Security and Cryptology – ICISC 2002*, Lecture Notes in Computer Science, Seoul, Korea, 28–29 November 2002. Springer-Verlag.

[MK00] Olivier Markowitch and Steve Kremer. A multi-party optimistic non-repudiation protocol. In *Information Security and Cryptology – ICISC 2000*, volume 2015 of *Lecture Notes in Computer Science*, pages 109–122, Seoul, Korea, December 2000. Springer-Verlag.

[MK01] Olivier Markowitch and Steve Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In *Information Security – ISC 2001*, volume 2200 of *Lecture Notes in Computer Science*, pages 363–378, Malaga, Spain, October 2001. Springer-Verlag.

[MR99] Olivier Markowitch and Yves Roggeman. Probabilistic non-repudiation without trusted third party. Presented at the Second Conference on Security in Communication Networks (SCN99), Amalfi, Italy, September 1999. No conference proceedings.

[MS01] Oliver Markowitch and Shahrokh Saeednia. Optimistic fair exchange with transparent signature recovery. In *Financial Cryptography – FC 2001*, volume 2339 of *Lecture Notes in Computer Science*, pages 339–350, Grand Cayman, British West Indies, 19–22 February 2001. Springer-Verlag.

[PG99] Henning Pagnia and Felix C. Gärtner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Department of Computer Science, Darmstadt, Germany, March 1999.

[PS96] Birgit Pfitzmann and Matthias Schunter. Asymmetric fingerprinting. In *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 84–95. Springer-Verlag, 1996.

[PSW98] Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Optimal efficiency of optimistic contract signing. In *Proceedings of the 17th Symposium on Principles of Distributed Computing (PODC '98)*, pages 113–122, New York, 1998. ACM Press.

[PV99] Henning Pagnia and Holger Vogt. Exchanging goods and payment in electronic business transactions. In *Proceedings of the Third European Research Seminar on Advances in Distributed Systems (ERSADS)*, Madeira Island, Portugal, April 1999.

[PVG02] Henning Pagnia, Holger Vogt, and Felix C. Gärtner. Fairer Austausch im Mobile Business. In Ralf Reichwald, editor, *Mobile Kommunikation – Wertschöpfung, Technologien, neue Dienste*, pages 157–170. Gabler Verlag, July 2002.

[PVG03] Henning Pagnia, Holger Vogt, and Felix C. Gärtner. Fair exchange. *The Computer Journal*, 46(1):55–75, January 2003.

[PVGW00] Henning Pagnia, Holger Vogt, Felix C. Gärtner, and Uwe G. Wilhelm. Solving fair exchange with mobile agents. In *ASA/MA 2000*, volume 1882 of *Lecture Notes in Computer Science*, pages 57–72, Zürich, Switzerland, September 2000. Springer-Verlag.

[PW97] Birgit Pfitzmann and Michael Waidner. Anonymous fingerprinting. In *Advances in Cryptology – EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 88–102. Springer-Verlag, 1997.

[Rab83] Michael O. Rabin. Transaction protection by beacons. *Journal of Computer and System Science*, 27:256–267, 1983.

[Rea02] Readnotify. http://www.readnotify.com, 2002.

[San97] Tuomas W. Sandholm. Unenforced e-commerce transactions. *IEEE Internet Computing*, 1(6):47–54, 1997.

[Sch93] Fred B. Schneider. What good are models and what models are good? In Sape Mullender, editor, *Distributed Systems*, chapter 2, pages 17–26. Addison-Wesley, second edition, 1993.

[Sch97] Berry Schoenmakers. Security aspects of the ecash payment system. In *COSIC '97 Course*, volume 1528 of *Lecture Notes in Computer Science*, pages 338–352, Leuven, Belgium, June 1997. Springer-Verlag.

[Sch00] Matthias Schunter. *Optimistic Fair Exchange*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, October 2000.

[SEM99] Advanced services, architecture and design. SEMPER Deliverable D10; La Gaude, March 1999. Available at http://www.semper.org/deliver/d10/d10.ps.gz.

[SL95] Thomas W. Sandholm and Victor R. Lesser. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 694–703, Montreal, Canada, August 20–25 1995. Morgan Kaufmann, San Mateo, CA.

[SM99] Vitaly Shmatikov and John C. Mitchell. Analysis of a fair exchange protocol. In *Proceedings of the 1999 FLoC Workshop*, Trento, Italy, 5 July 1999.

[Syv98] Paul Syverson. Weakly secret bit commitment: Applications to lotteries and fair exchange. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pages 2–13, Rockport, Massachusetts, June 1998. IEEE Computer Society Press.

[Ted85] Tom Tedrick. Fair exchange of secrets. In *Advances in Cryptology – CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 434–438. Springer-Verlag, 1985.

[Tyg96] J. D. Tygar. Atomicity in electronic commerce. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC '96)*, pages 8–26, Philadelphia, PA, May 1996. ACM Press.

[Tyg98] J. D. Tygar. Atomicity versus anonymity: Distributed transactions for electronic commerce. In Ashish Gupta, Oded Shmueli, and Jennifer Widom, editors, *Proceedings of the 24rd International Conference on Very Large Data Bases – VLDB '98*, pages 1–12, New York, USA, 24–27 August 1998. Morgan Kaufmann.

[VGP03] Holger Vogt, Felix C. Gärtner, and Henning Pagnia. Supporting fair exchange in mobile environments. *ACM/Kluwer Journal on Mobile Networks and Applications (MONET)*, 8(2):127–136, April 2003.

[Vog03] Holger Vogt. Asynchronous optimistic fair exchange based on revocable items. In *Financial Cryptography – FC 2003*, volume 2742 of *Lecture Notes in Computer Science*, pages 208–222, Guadeloupe, French West Indies, 27–30 January 2003. Springer-Verlag.

[VP99] Holger Vogt and Henning Pagnia. Fairer Austausch beim elektronischen Einkauf im Internet. In *Proceedings of the 6th DFN-CERT Workshop "Sicherheit in vernetzten Systemen"*, Hamburg, Germany, March 1999.

[VPG99] Holger Vogt, Henning Pagnia, and Felix C. Gärtner. Modular fair exchange protocols for electronic commerce. In *Proceedings of the 15th Annual Computer Security Applications Conference*, pages 3–11, Phoenix, Arizona, December 1999. IEEE Computer Society Press.

[VPG01] Holger Vogt, Henning Pagnia, and Felix C. Gärtner. Using smart cards for fair exchange. In *Electronic Commerce – WELCOM 2001*, volume 2232 of *Lecture Notes in Computer Science*, pages 101–113, Heidelberg, Germany, 16–17 November 2001. Springer-Verlag.

[WV01] Chuan-Kun Wu and Vijay Varadharajan. Fair exchange of digital signatures with offline trusted third party. In *Information and Communications Security – ICICS 2001*, volume 2229 of *Lecture Notes in Computer Science*, pages 466–470, Xian, China, November 2001. Springer-Verlag.

[ZDB99] Jianying Zhou, Robert Deng, and Feng Bao. Evolution of fair non-repudiation with TTP. In *Information Security and Privacy – ACISP '99*, volume 1587 of *Lecture Notes in Computer Science*, pages 258–269, Wollongong, Australia, 7–9 April 1999. Springer-Verlag.

[ZDB00] Jianying Zhou, Robert Deng, and Feng Bao. Some remarks on a fair exchange protocol. In *Public Key Cryptography – PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 46–57, Melbourne, Australia, January 2000. Springer-Verlag.

[ZG96] Jianying Zhou and Dieter Gollmann. A fair non-repudiation protocol. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, CA, May 1996. IEEE Computer Society Press.

[ZG97] Jianying Zhou and Dieter Gollmann. An efficient non-repudiation protocol. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop*, pages 126–132, Rockport, MA, June 1997. IEEE Computer Society Press.

[ZL99] Jianying Zhou and Kwok-Yan Lam. A secure pay-per-view scheme for web-based video service. In *Public Key Cryptography – PKC '99*, volume 1560 of *Lecture Notes in Computer Science*, pages 315–326, Kamakura, Japan, March 1999. Springer-Verlag.

Index